MOTOROLA
intelligence everywhere™

digital dna™

MPC8560xxxx
1/2003
Rev. 3.0

MPC8560 Power QUICC III
Integrated Communications 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.

© Motorola, Inc. 2002

Contents

Paragraph Number	Title	Page Number
	Part I - Overview	
	Overview	
0.1	Introduction	1-1
0.2	MPC8560 Overview	1-1
0.2.1	Key Features	1-2
0.3	MPC8560 Architecture Overview	1-9
0.3.1	e500 Core Overview	1-9
0.3.2	Communications Processor Module (CPM)	1-13
0.3.3	On-Chip Memory Unit	1-15
0.3.3.1	On-Chip Memory as Memory-Mapped SRAM	1-15
0.3.3.2	On-Chip Memory as L2 Cache	1-15
0.3.4	e500 Coherency Module (ECM)	1-16
0.3.5	DDR SDRAM Controller	1-16
0.3.6	Programmable Interrupt Controller (PIC)	1-17
0.3.7	I2C Controller	1-17
0.3.8	Boot Sequencer	1-18
0.3.9	Local Bus Controller	1-18
0.3.10	Three Speed Ethernet Controllers (10/100/1000)	1-19
0.3.11	Integrated DMA	1-19
0.3.12	PCI Controller	1-19
0.3.13	RapidIO Controller	1-20
0.3.13.1	RapidIO Message Unit	1-20
0.3.14	Power Management	1-21
0.3.15	Clocking	1-21
0.3.16	Address Map	1-21
0.3.17	OceaN Switch Fabric	1-22
0.4	Data Processing Overview	1-22
0.5	Processing Between the CPM and Local Bus	1-22
0.6	Processing Across the On-Chip Fabric	1-23
0.7	Data Processing with the e500 Coherency Module	1-24
0.8	MPC8560 Application Examples	1-25
0.8.1	Device Configurations	1-25
0.8.1.1	Single-Processor System	1-25
0.8.1.2	Multiprocessor System	1-26

Contents

Paragraph Number	Title	Page Number
0.8.1.3	High-Performance System	1-27
0.8.2	Examples of Communications Systems	1-27
0.8.2.1	Remote Access Server	1-28
0.8.2.2	Regional Office Router	1-29
0.8.2.3	LAN-to-WAN Bridge Router	1-30
0.8.2.4	Cellular Base Station	1-31
0.8.2.5	3G Wireless Base Station	1-32
0.8.2.6	Telecommunications Switch Controller	1-33
0.8.2.7	SONET Transmission Controller	1-34
0.8.2.8	Frame Relay Card	1-35
0.8.2.9	ATM Protocol Converter	1-36
0.9	Compatibility Issues	1-36
0.9.1	Software	1-36
0.9.2	MPC8560 Hardware	1-36
0.9.3	Differences Between MPC8560 and MPC8260	1-37
0.9.4	Communications Protocol Table	1-37
0.9.5	MPC8560 Configurations	1-37
0.9.6	Pin Configurations	1-38
0.9.7	Communications Performance	1-38

Chapter 1
Memory Map

1.1	Local Memory Map Overview and Example	2-1
1.2	Address Translation and Mapping	2-3
1.2.1	SRAM Windows	2-4
1.2.2	Window into Configuration Space	2-4
1.2.3	Local Access Windows	2-4
1.2.3.1	Local Access Register Memory Map	2-5
1.2.3.2	Local Access Window x Base Address Registers (LAWBAR0–LAWBAR7)	2-6
1.2.3.3	Local Access Window **x** Attributes Registers (LAWAR0–LAWAR7)	2-7
1.2.3.4	Precedence of Local Access Windows	2-8
1.2.3.5	Configuring Local Access Windows	2-8
1.2.3.6	Distinguishing Local Access Windows from Other Mapping Functions	2-8
1.2.3.7	Illegal Interaction between Local Access Windows and DDR SDRAM Chip Selects	2-9
1.2.4	Outbound Address Translation and Mapping Windows	2-9
1.2.5	Inbound Address Translation and Mapping Windows	2-9
1.2.5.1	RapidIO Inbound ATMU	2-9
1.2.5.2	PCI/PCI-X Inbound ATMU	2-10
1.2.5.3	Illegal Interaction between Inbound ATMUs and Local Access Windows	2-10

Contents

Paragraph Number	Title	Page Number
1.3	Configuration, Control, and Status Register Map	2-10
1.3.1	Accessing CCSR Memory from External Masters	2-11
1.3.2	Organization of CCSR Memory	2-12
1.3.3	General Utilities Registers	2-12
1.3.4	Interrupt Controller and CCSR	2-14
1.3.5	Communications Processor Module and CCSR	2-14
1.3.6	RapidIO and CCSR	2-15
1.3.7	Device-Specific Utilities	2-16
1.4	Complete CCSR Map	2-17

Chapter 2
Signal Descriptions

2.1	Introduction	3-1
2.2	Signals Overview	3-1
2.3	Configuration Signals Sampled at Reset	3-14
2.4	Output Signal States During Reset	3-15

Chapter 3
Reset, Clocking, and Initialization

3.1	Introduction	4-1
3.1.1	Overview	4-1
3.2	External Signal Description	4-1
3.2.1	Detailed Signal Descriptions	4-1
3.2.1.1	System Control Signals	4-1
3.2.1.2	Clock Signals	4-2
3.3	Memory Map/Register Definition	4-3
3.4	Functional Description	4-3
3.4.1	Reset Operations	4-3
3.4.1.1	Hard Reset	4-3
3.4.1.2	Soft Reset	4-3
3.4.2	Power-On Reset Sequence	4-3
3.4.3	Power-On Reset (POR) Configuration	4-5
3.4.3.1	System PLL Ratio	4-6
3.4.3.2	e500 Core PLL Ratio	4-6
3.4.3.3	Boot ROM Location	4-7
3.4.3.4	Host/Agent Configuration	4-7
3.4.3.5	CPU Boot Configuration	4-8
3.4.3.6	Boot Sequencer Configuration	4-8
3.4.3.7	TSEC Width	4-9
3.4.3.8	TSEC1 Protocol	4-9

Contents

3.4.3.9 TSEC2 Protocol 4-10
3.4.3.10 RapidIO Transmit Clock Source 4-10
3.4.3.11 RapidIO Device ID 4-10
3.4.3.12 PCI Width Configuration 4-11
3.4.3.13 PCI I/O Impedance 4-11
3.4.3.14 PCI Arbiter Configuration 4-11
3.4.3.15 PCI Debug Configuration 4-12
3.4.3.16 PCI-X Configuration 4-12
3.4.3.17 Memory Debug Configuration 4-12
3.4.3.18 DDR Debug Configuration 4-13
3.4.3.19 PCI/PCI-X Output Hold Configuration 4-13
3.4.3.20 Local Bus Output Hold Configuration 4-14
3.4.4 Clocking 4-14
3.4.4.1 System Clock/PCI Clock 4-14
3.4.4.2 RapidIO Clocks 4-15
3.4.4.3 Ethernet Clocks 4-16
3.4.4.4 Real Time Clock 4-16
3.5 Initialization/Applications Information 4-16
3.5.1 Local Configuration Control 4-16
3.5.1.1 Accessing Configuration, Control, and Status Registers 4-17
3.5.1.1.1 Updating CCSRBAR 4-17
3.5.1.1.2 Configuration Control and Status Register Base Address Register (CCSRBAR) 4-18
3.5.1.2 Accessing Alternate Configuration Space 4-18
3.5.1.2.1 Alternate Configuration Base Address Register (ALTCBAR) 4-19
3.5.1.2.2 Alternate Configuration Attribute Register (ALTCAR) 4-20
3.5.1.3 Boot Page Translation 4-20
3.5.1.3.1 Boot Page Translation Register (BPTR) 4-21
3.5.2 Boot Sequencer 4-21

Part II - e500 Core Complex and L2 Cache

Chapter 4
e500 Core Complex Overview

4.1 Overview 5-1
4.2 Features 5-5
4.3 Instruction Set 5-11
4.4 Instruction Flow 5-13
4.4.1 Branch Detection and Prediction 5-13
4.4.2 e500 Execution Pipeline 5-14
4.5 Programming Model 5-17

Contents

Paragraph Number	Title	Page Number
4.6	On-Chip Cache Implementation	5-19
4.7	Interrupts and Exception Handling	5-19
4.7.1	Exception Handling	5-19
4.7.2	Interrupt Classes	5-20
4.7.3	Interrupt Types	5-20
4.7.4	Upper Bound on Interrupt Latencies	5-21
4.7.5	Interrupt Registers	5-22
4.8	Memory Management	5-24
4.8.1	Address Translation	5-26
4.8.2	MMU Assist Registers (MAS1–MAS4 and MAS6)	5-26
4.8.3	Process ID Registers (PID0–PID2)	5-27
4.8.4	TLB Coherency	5-27
4.9	Memory Coherency	5-27
4.9.1	Atomic Update Memory References	5-28
4.9.2	Memory Access Ordering	5-28
4.9.3	Cache Control Instructions	5-28
4.9.4	Programmable Page Characteristics	5-28
4.10	Core Complex Bus (CCB)	5-29
4.11	Performance Monitoring	5-29
4.11.1	Global Control Register	5-29
4.11.2	Performance Monitor Counter Registers	5-30
4.11.3	Local Control Registers	5-30
4.12	Legacy Support of PowerPC Architecture	5-30
4.12.1	Instruction Set Compatibility	5-31
4.12.1.1	User Instruction Set	5-31
4.12.1.2	Supervisor Instruction Set	5-31
4.12.2	Memory Subsystem	5-31
4.12.3	Exception Handling	5-31
4.12.4	Memory Management	5-32
4.12.5	Reset	5-32
4.12.6	Other Differences	5-32

Chapter 5
e500 Register Summary

5.1	e500 Register Model	6-1
5.2	Register Model for 32-Bit Book E Implementations	6-3
5.2.1	Special-Purpose Registers (SPRs)	6-4
5.3	Registers for Integer Operations	6-8
5.3.1	General Purpose Registers (GPRs)	6-8
5.3.2	Integer Exception Register (XER)	6-8
5.4	Registers for Branch Operations	6-8

Contents

Paragraph Number	Title	Page Number
5.4.1	Condition Register (CR)	6-9
5.4.2	Link Register (LR)	6-10
5.4.3	Count Register (CTR)	6-10
5.5	Processor Control Registers	6-10
5.5.1	Machine State Register (MSR)	6-10
5.5.2	Processor ID Register (PIR)	6-13
5.5.3	Processor Version Register (PVR)	6-13
5.6	Timer Registers	6-13
5.6.1	Timer Control Register (TCR)	6-13
5.6.2	Timer Status Register (TSR)	6-14
5.6.3	Time Base	6-15
5.6.4	Decrementer Register	6-16
5.6.5	Decrementer Auto-Reload Register (DECAR)	6-16
5.7	Interrupt Registers	6-16
5.7.1	Interrupt Registers Defined by Book E	6-16
5.7.1.1	Save/Restore Register 0 (SRR0)	6-16
5.7.1.2	Save/Restore Register 1 (SRR1)	6-16
5.7.1.3	Critical Save/Restore Register 0 (CSRR0)	6-17
5.7.1.4	Critical Save/Restore Register 1 (CSRR1)	6-17
5.7.1.5	Data Exception Address Register (DEAR)	6-17
5.7.1.6	Interrupt Vector Prefix Register (IVPR)	6-17
5.7.1.7	Interrupt Vector Offset Registers (IVORs)	6-18
5.7.1.8	Exception Syndrome Register (ESR)	6-19
5.7.2	e500-Specific Interrupt Registers	6-20
5.7.2.1	Machine Check Save/Restore Register 0 (MCSRR0)	6-20
5.7.2.2	Machine Check Save/Restore Register 1 (MCSRR1)	6-20
5.7.2.3	Machine Check Syndrome Register (MCSR)	6-21
5.8	Software-Use SPRs (SPRG0–SPRG7 and USPRG0)	6-22
5.9	Branch Target Buffer (BTB) Registers	6-23
5.9.1	Branch Buffer Entry Address Register (BBEAR)	6-23
5.9.2	Branch Buffer Target Address Register (BBTAR)	6-23
5.9.3	Branch Unit Control and Status Register (BUCSR)	6-24
5.10	Hardware Implementation-Dependent Registers	6-24
5.10.1	Hardware Implementation-Dependent Register (HID0)	6-24
5.10.2	Hardware Implementation-Dependent Register 1 (HID1)	6-25
5.11	L1 Cache Configuration Registers	6-27
5.11.1	L1 Cache Control and Status Register 0 (L1CSR0)	6-27
5.11.2	L1 Cache Control and Status Register 1 (L1CSR1)	6-28
5.11.3	L1 Cache Configuration Register 0 (L1CFG0)	6-29
5.11.4	L1 Cache Configuration Register 1 (L1CFG1)	6-30
5.12	MMU Registers	6-31

Contents

Paragraph Number	Title	Page Number
5.12.1	Process ID Registers (PID0–PID2)	6-31
5.12.2	MMU Control and Status Register 0 (MMUCSR0)	6-31
5.12.3	MMU Configuration Register (MMUCFG)	6-32
5.12.4	TLB Configuration Registers (TLBnCFG)	6-33
5.12.4.1	TLB0 Configuration Register (TLB0CFG)	6-33
5.12.4.2	TLB Configuration Register 1 (TLB1CFG)	6-33
5.12.5	MMU Assist Registers (MAS0–MAS4, MAS6)	6-34
5.12.5.1	MAS Register 0 (MAS0)	6-34
5.12.5.2	MAS Register 1 (MAS1)	6-35
5.12.5.3	MAS Register 2 (MAS2)	6-36
5.12.5.4	MAS Register 3 (MAS3)	6-37
5.12.5.5	MAS Register 4 (MAS4)	6-38
5.12.5.6	MAS Register 6 (MAS6)	6-39
5.13	Debug Registers	6-39
5.13.1	Debug Control Registers	6-39
5.13.1.1	Debug Control Register 0 (DBCR0)	6-39
5.13.1.2	Debug Control Register 1	6-41
5.13.1.3	Debug Control Register 2 (DBCR2)	6-43
5.13.2	Debug Status Register (DBSR)	6-45
5.13.3	Instruction Address Compare Registers (IAC1–IAC2)	6-46
5.13.4	Data Address Compare Registers (DAC1–DAC2)	6-46
5.14	Signal Processing Engine (SPE) APU Registers	6-47
5.14.1	Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)	6-47
5.14.2	Accumulator (ACC)	6-48
5.15	Performance Monitor Registers	6-49
5.15.1	Global Control Register 0 (PMGC0)	6-50
5.15.2	User Global Control Register 0 (UPMGC0)	6-51
5.15.3	Local Control A Registers (PMLCa0–PMLCa3)	6-51
5.15.4	User Local Control A Registers (UPMLCa0–UPMLCa3)	6-52
5.15.5	Local Control B Registers (PMLCb0–PMLCb3)	6-52
5.15.6	User Local Control B Registers (UPMLCb0–UPMLCb3)	6-53
5.15.7	Performance Monitor Counter Registers (PMC0–PMC3)	6-53

Chapter 6
L2 Look-Aside Cache/SRAM

6.1	L2 Cache Overview	7-1
6.1.1	L2 Cache/SRAM Features	7-2
6.2	Cache Organization	7-2
6.3	Memory Map/Register Definition	7-5
6.3.1	L2/SRAM Register Descriptions	7-6

Contents

Paragraph Number	Title	Page Number
6.3.1.1	L2 Control Register (L2CTL)	7-6
6.3.1.2	L2 Cache External Write Address Registers (L2CEWAR*n*)	7-9
6.3.1.3	L2 Cache External Write Control Registers (L2CEWCR0–L2CEWCR03)	7-10
6.3.1.4	L2 Memory-Mapped SRAM Base Address Registers (L2SRBAR*n*)	7-11
6.3.1.5	L2 Error Registers	7-12
6.3.1.5.1	Error Injection Registers	7-12
6.3.1.5.2	Error Control and Capture Registers	7-14
6.4	L2 Cache Timing	7-21
6.5	SRAM Coherency	7-22
6.5.1	L2 Cache Coherency Rules	7-22
6.5.2	Memory-Mapped SRAM Coherency Rules	7-23
6.6	L2 Cache Locking	7-23
6.6.1	Locking the Entire L2 Cache	7-23
6.6.2	Locking Programmed Memory Ranges	7-24
6.6.3	Locking Selected Lines	7-24
6.6.4	Clearing Locks On Selected Lines	7-24
6.6.5	Flash Clearing of Instruction and Data Locks	7-25
6.6.6	Locks With Stale Data	7-25
6.7	L2 Replacement Policy	7-25
6.7.1	PLRU Bit Update Considerations	7-26
6.7.2	Allocation of Lines	7-27
6.8	L2 Cache Operation	7-27
6.8.1	L2 Cache States	7-27
6.8.2	L2 State Transitions	7-28

Part III - Memory and I/O Interfaces

Chapter 7
e500 Coherency Module

7.1	Introduction	8-1
7.2	Overview	8-1
7.2.1	Features	8-2
7.3	Memory Map/Register Definition	8-2
7.3.1	Register Descriptions	8-3
7.3.1.1	ECM CCB Address Configuration Register (EEBACR)	8-3
7.3.1.2	ECM CCB Port Configuration Register (EEBPCR)	8-4
7.3.1.3	ECM Error Detect Register (EEDR)	8-5
7.3.1.4	ECM Error Enable Register (EEER)	8-5
7.3.1.5	ECM Error Attributes Capture Register (EEATR)	8-6
7.3.1.6	ECM Error Address Capture Register (EEADR)	8-8

Contents

Paragraph Number	Title	Page Number
7.4	Functional Description	8-9
7.4.1	I/O Arbiter	8-9
7.4.2	CCB Arbiter	8-9
7.4.3	Transaction Queue	8-9
7.4.4	Global Data Multiplexor	8-9
7.4.5	CCB Interface	8-9
7.5	Initialization/Application Information	8-10

Chapter 8
DDR Memory Controller

8.1	Features	9-2
8.1.1	Modes of Operation	9-3
8.2	External Signal Description	9-3
8.2.1	Signals Overview	9-3
8.2.2	Detailed Signal Descriptions	9-5
8.2.2.1	Memory Interface Signals	9-5
8.2.2.2	Clock Interface Signals	9-9
8.2.2.3	Debug Signals	9-10
8.3	Memory Map/Register Definition	9-10
8.3.1	Register Descriptions	9-11
8.3.1.1	Chip Select Memory Bounds (CS*n*_BNDS)	9-11
8.3.1.2	Chip Select Configuration (CS*n*_CONFIG)	9-12
8.3.1.3	DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)	9-13
8.3.1.4	DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)	9-16
8.3.1.5	DDR SDRAM Control Configuration (DDR_SDRAM_CFG)	9-17
8.3.1.6	DDR SDRAM Mode Configuration (DDR_SDRAM_MODE)	9-18
8.3.1.7	DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)	9-19
8.3.1.8	Memory Data Path Error Injection Mask High (DATA_ERR_INJECT_HI)	9-20
8.3.1.9	Memory Data Path Error Injection Mask Low (DATA_ERR_INJECT_LO)	9-21
8.3.1.10	Memory Data Path Error Injection Mask ECC (ECC_ERR_INJECT)	9-21
8.3.1.11	Memory Data Path Read Capture High (CAPTURE_DATA_HI)	9-22
8.3.1.12	Memory Data Path Read Capture Low (CAPTURE_DATA_LO)	9-23
8.3.1.13	Memory Data Path Read Capture ECC (CAPTURE_ECC)	9-23
8.3.1.14	Memory Error Detect (ERR_DETECT)	9-24
8.3.1.15	Memory Error Disable (ERR_DISABLE)	9-25
8.3.1.16	Memory Error Interrupt Enable (ERR_INT_EN)	9-25
8.3.1.17	Memory Error Attributes Capture (CAPTURE_ATTRIBUTES)	9-26
8.3.1.18	Memory Error Address Capture (CAPTURE_ADDRESS)	9-28

Contents

Paragraph Number	Title	Page Number
8.3.1.19	Single-Bit ECC Memory Error Management (ERR_SBE)	9-29
8.4	Functional Description	9-30
8.4.1	DDR SDRAM Interface Operation	9-35
8.4.1.1	Supported DDR SDRAM Organizations	9-36
8.4.2	DDR SDRAM Address Multiplexing	9-37
8.4.3	JEDEC Standard DDR SDRAM Interface Commands	9-38
8.4.4	SDRAM Interface Timing	9-39
8.4.4.1	Clock Distribution	9-43
8.4.5	DDR SDRAM Mode-Set Command Timing	9-43
8.4.6	DDR SDRAM Registered DIMM Mode	9-44
8.4.7	DDR SDRAM Write Timing Adjustments	9-45
8.4.8	DDR SDRAM Refresh	9-46
8.4.8.1	DDR SDRAM Refresh Timing	9-47
8.4.8.2	DDR SDRAM Refresh and Power-Saving Modes	9-48
8.4.8.2.1	Self-Refresh in Sleep Modes	9-48
8.4.9	DDR Data Beat Ordering	9-50
8.4.10	Page Mode and Logical Bank Retention	9-50
8.4.11	Error Checking and Correcting (ECC)	9-51
8.4.12	Error Management	9-53
8.5	Initialization/Application Information	9-54
8.5.1	DDR SDRAM Initialization Sequence	9-55

Chapter 9
Programmable Interrupt Controller

9.1	Introduction	10-1
9.1.1	Overview	10-4
9.1.2	Features	10-4
9.1.3	Modes of Operation	10-5
9.1.3.1	Mixed Mode (GCR[M] Set)	10-5
9.1.3.2	Pass-Through Mode (GCR[M] Cleared)	10-5
9.1.4	Internal Interrupt Sources	10-6
9.2	External Signal Description	10-7
9.2.1	Signal Overview	10-7
9.2.2	Detailed Signal Descriptions	10-7
9.3	Memory Map/Register Definition	10-9
9.3.1	Global Registers	10-15
9.3.1.1	Feature Reporting Register (FRR)	10-15
9.3.1.2	Global Configuration Register (GCR)	10-16
9.3.1.3	Vendor Identification Register (VIR)	10-16
9.3.1.4	Processor Initialization Register (PIR)	10-17
9.3.1.5	IPI Vector/Priority Registers (IPIVPR*n*)	10-18

Contents

Paragraph Number	Title	Page Number
9.3.1.6	Spurious Vector Register (SVR)	10-19
9.3.2	Global Timer Registers	10-20
9.3.2.1	Timer Frequency Reporting Register (TFRR)	10-20
9.3.2.2	Global Timer Current Count Registers (GTCCR*n*)	10-20
9.3.2.3	Global Timer Base Count Registers (GTBCR*n*)	10-21
9.3.2.4	Global Timer Vector/Priority Registers (GTVPR*n*)	10-22
9.3.2.5	Global Timer Destination Registers (GTDR)	10-23
9.3.2.6	Timer Control Register (TCR)	10-24
9.3.3	Summary Registers	10-26
9.3.3.1	$\overline{\text{IRQ_OUT}}$ Summary Register 0 (IRQSR0)	10-26
9.3.3.2	$\overline{\text{IRQ_OUT}}$ Summary Register 1 (IRQSR1)	10-27
9.3.3.3	Critical Interrupt Summary Register 0 (CISR0)	10-27
9.3.3.4	Critical Interrupt Summary Register 1 (CISR1)	10-28
9.3.4	Performance Monitor Mask Registers (PMMRs)	10-29
9.3.4.1	Performance Monitor Mask Register (Even) (PMMR0/2/4/6)	10-29
9.3.4.2	Performance Monitor Mask Registers (Odd) (PMMR1/3/5/7)	10-30
9.3.5	Message Registers	10-30
9.3.5.1	Message Registers (MSGR0–MSGR3)	10-31
9.3.5.2	Message Enable Register (MER)	10-31
9.3.5.3	Message Status Register (MSR)	10-32
9.3.6	Interrupt Source Configuration Registers	10-33
9.3.6.1	External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)	10-33
9.3.6.2	External Interrupt Destination Registers (EIDR0–EIDR11)	10-34
9.3.6.3	Internal Interrupt Vector/Priority Registers (IIVPR0–IIVPR31)	10-35
9.3.6.4	Internal Interrupt Destination Registers (IIDR0–IIDR31)	10-36
9.3.6.5	Messaging Interrupt Vector/Priority Registers (MIVPR0–MIVPR3)	10-37
9.3.6.6	Messaging Interrupt Destination Registers (MIDR0–MIDR3)	10-38
9.3.7	Per CPU Registers	10-39
9.3.7.1	Interprocessor Interrupt Dispatch Register (IPIDR0–IPIDR3)	10-41
9.3.7.2	Processor Current Task Priority Register (CTPR)	10-42
9.3.7.3	Who Am I Register (WHOAMI)	10-43
9.3.7.4	Processor Interrupt Acknowledge Register (IACK)	10-44
9.3.7.5	Processor End of Interrupt Register (EOI)	10-45
9.4	Functional Description	10-45
9.4.1	Interrupt Source Priority	10-46
9.4.2	Processor Current Task Priority	10-47
9.4.3	Interrupt Acknowledge	10-47
9.4.4	Nesting of Interrupts	10-47
9.4.5	Processor Initialization	10-47
9.4.6	Spurious Vector Generation	10-48
9.4.7	Messaging Interrupts	10-48

Contents

Paragraph Number	Title	Page Number
9.4.8	Timers	10-48
9.4.9	Reset	10-49
9.5	Initialization/Application Information	10-49
9.5.1	Programming Guidelines	10-49
9.5.1.1	PIC Registers	10-49
9.5.1.2	Changing Interrupt Source Configuration	10-51
9.5.1.3	MSG, IPI, and Timer Priority Levels	10-51

Chapter 10
I2C Interface

10.1	Overview	11-1
10.1.1	Features	11-2
10.1.2	Modes of Operation	11-3
10.2	External Signal Description	11-3
10.2.1	Signal Overview	11-3
10.2.2	Detailed Signal Descriptions	11-4
10.3	Memory Map/Register Definition	11-5
10.3.1	Register Descriptions	11-5
10.3.1.1	I2C Address Register (I2CADR)	11-5
10.3.1.2	I2C Frequency Divider Register (I2CFDR)	11-6
10.3.1.3	I2C Control Register (I2CCR)	11-7
10.3.1.4	I2C Status Register (I2CSR)	11-9
10.3.1.5	I2C Data Register (I2CDR)	11-10
10.3.1.6	Digital Filter Sampling Rate Register (I2CDFSRR)	11-11
10.4	Functional Description	11-11
10.4.1	START Condition	11-12
10.4.2	Slave Address Transmission	11-12
10.4.3	Repeated START Condition	11-13
10.4.4	STOP Condition	11-13
10.4.5	Arbitration Procedure	11-14
10.4.6	Clock Synchronization	11-14
10.4.7	Handshaking	11-15
10.4.8	Clock Stretching	11-15
10.4.9	Description of I^2C Functional Blocks	11-15
10.4.9.1	Clock Control	11-15
10.4.9.2	Input Synchronization and Digital Filter	11-16
10.4.9.2.1	Input Signal Synchronization	11-16
10.4.9.2.2	Filtering of SCL and SDA Lines	11-16
10.4.9.3	Transaction Monitoring	11-16
10.4.9.4	Arbitration Control	11-16
10.4.9.5	Control Transfer	11-17

Contents

Paragraph Number	Title	Page Number
10.4.9.6	Address Compare	11-18
10.4.10	Boot Sequencer Mode	11-18
10.4.10.1	EEPROM Calling Address	11-19
10.4.10.2	EEPROM Data Format	11-19
10.5	Initialization/Application Information	11-22
10.5.1	Initialization Sequence	11-22
10.5.2	Generation of START	11-22
10.5.3	Post-Transfer Software Response	11-23
10.5.4	Generation of STOP	11-23
10.5.5	Generation of Repeated START	11-24
10.5.6	Generation of SCL When SDA Low	11-24
10.5.7	Slave Mode Interrupt Service Routine	11-24
10.5.7.1	Slave Transmitter and Received Acknowledge	11-25
10.5.7.2	Loss of Arbitration and Forcing of Slave Mode	11-25
10.5.8	Interrupt Service Routine Flowchart	11-25

Chapter 11
Local Bus Controller

11.1	Introduction	12-1
11.1.1	Overview	12-2
11.1.2	Features	12-2
11.1.3	Modes of Operation	12-3
11.1.3.1	LBC Bus Clock and Clock Ratios	12-4
11.1.3.2	Source ID Debug Mode	12-4
11.1.4	Power-Down Mode	12-4
11.1.5	References	12-4
11.2	External Signal Description	12-5
11.3	Memory Map/Register Definition	12-10
11.3.1	Register Descriptions	12-11
11.3.1.1	Base Registers (BR0–BR7)	12-11
11.3.1.2	Option Registers (OR0–OR7)	12-13
11.3.1.2.1	Address Mask	12-13
11.3.1.2.2	Option Registers (ORx)—GPCM Mode	12-14
11.3.1.2.3	Option Registers (ORx)—UPM Mode	12-17
11.3.1.2.4	Option Registers (ORx)—SDRAM Mode	12-18
11.3.1.3	UPM Memory Address Register (MAR)	12-20
11.3.1.4	UPM Mode Registers (MxMR)	12-20
11.3.1.5	Memory Refresh Timer Prescaler Register (MRTPR)	12-23
11.3.1.6	UPM Data Register (MDR)	12-24
11.3.1.7	SDRAM Machine Mode Register (LSDMR)	12-25
11.3.1.8	UPM Refresh Timer (LURT)	12-27

Contents

Paragraph Number	Title	Page Number
11.3.1.9	SDRAM Refresh Timer (LSRT)	12-28
11.3.1.10	Transfer Error Status Register (LTESR)	12-29
11.3.1.11	Transfer Error Check Disable Register (LTEDR)	12-31
11.3.1.12	Transfer Error Interrupt Enable Register (LTEIR)	12-32
11.3.1.13	Transfer Error Attributes Register (LTEATR)	12-33
11.3.1.14	Transfer Error Address Register (LTEAR)	12-34
11.3.1.15	Local Bus Configuration Register (LBCR)	12-34
11.3.1.16	Clock Ratio Register (LCRR)	12-35
11.4	Functional Description	12-37
11.4.1	Basic Architecture	12-38
11.4.1.1	Address and Address Space Checking	12-38
11.4.1.2	External Address Latch Enable Signal (LALE)	12-38
11.4.1.3	Data Transfer Acknowledge (TA)	12-40
11.4.1.4	Data Buffer Control (LBCTL)	12-41
11.4.1.5	Atomic Operation	12-41
11.4.1.6	Parity Generation and Checking (LDP)	12-42
11.4.1.7	Bus Monitor	12-42
11.4.2	General-Purpose Chip-Select Machine (GPCM)	12-42
11.4.2.1	Timing Configuration	12-43
11.4.2.2	Chip-Select Assertion Timing	12-47
11.4.2.2.1	Programmable Wait State Configuration	12-48
11.4.2.2.2	Chip-Select and Write Enable Negation Timing	12-48
11.4.2.2.3	Relaxed Timing	12-49
11.4.2.2.4	Output Enable (LOE) Timing	12-52
11.4.2.2.5	Extended Hold Time on Read Accesses	12-52
11.4.2.3	External Access Termination (LGTA)	12-54
11.4.2.4	Boot Chip-Select Operation	12-54
11.4.3	SDRAM Machine	12-55
11.4.3.1	Supported SDRAM Configurations	12-55
11.4.3.2	SDRAM Power-On Initialization	12-56
11.4.3.3	Intel PC133 and JEDEC-Standard SDRAM Interface Commands	12-57
11.4.3.4	Page Hit Checking	12-58
11.4.3.5	Page Management	12-58
11.4.3.6	SDRAM Address Multiplexing	12-58
11.4.3.7	SDRAM Device-Specific Parameters	12-59
11.4.3.7.1	Precharge-to-Activate Interval	12-60
11.4.3.7.2	Activate to Read/Write Interval	12-60
11.4.3.7.3	Column Address to First Data Out—CAS Latency	12-60
11.4.3.7.4	Last Data In to Precharge—Write Recovery	12-61
11.4.3.7.5	Refresh Recovery Interval (RFRC)	12-61
11.4.3.7.6	External Address and Command Buffers (BUFCMD)	12-62

Contents

Paragraph Number	Title	Page Number
11.4.3.8	SDRAM Interface Timing	12-62
11.4.3.9	SDRAM Read/Write Transactions	12-64
11.4.3.10	SDRAM MODE-SET Command Timing	12-65
11.4.3.11	SDRAM Refresh	12-65
11.4.3.11.1	SDRAM Refresh Timing	12-65
11.4.4	User-Programmable Machines (UPMs)	12-66
11.4.4.1	UPM Requests	12-67
11.4.4.1.1	Memory Access Requests	12-68
11.4.4.1.2	UPM Refresh Timer Requests	12-69
11.4.4.1.3	Software Requests—RUN Command	12-69
11.4.4.1.4	Exception Requests	12-70
11.4.4.2	Programming the UPMs	12-70
11.4.4.3	UPM Signal Timing	12-70
11.4.4.4	The RAM Array	12-71
11.4.4.4.1	RAM Words	12-72
11.4.4.4.2	Chip-Select Signal Timing (CST*n*)	12-76
11.4.4.4.3	Byte-Select Signal Timing (BST*n*)	12-76
11.4.4.4.4	General-Purpose Signals (GxTx, GOx)	12-77
11.4.4.4.5	Loop Control (LOOP)	12-77
11.4.4.4.6	Repeat Execution of Current RAM Word (REDO)	12-78
11.4.4.4.7	Address Multiplexing (AMX)	12-78
11.4.4.4.8	Data Valid and Data Sample Control (UTA)	12-79
11.4.4.4.9	LGPL[0:5] Signal Negation (LAST)	12-80
11.4.4.4.10	The Wait Mechanism (WAEN)	12-80
11.4.4.5	Synchronous Sampling of UPWAIT for Early Transfer Acknowledge	12-81
11.4.4.6	Extended Hold Time on Read Accesses	12-82
11.4.4.7	Memory System Interface Example Using UPM	12-82
11.5	Initialization/Application Information	12-88
11.5.1	Interfacing to Peripherals	12-88
11.5.1.1	Multiplexed Address/Data Bus and Unmultiplexed Address Signals	12-88
11.5.1.2	Peripheral Hierarchy on the Local Bus	12-89
11.5.1.3	Peripheral Hierarchy on the Local Bus for Very High Bus Speeds	12-89
11.5.1.4	GPCM Timings	12-90
11.5.2	Bus Turnaround	12-91
11.5.2.1	Address Phase after Previous Read	12-91
11.5.2.2	Read Data Phase after Address Phase	12-92
11.5.2.3	Read-Modify-Write Cycle for Parity Protected Memory Banks	12-92
11.5.2.4	UPM Cycles with Additional Address Phases	12-92
11.5.3	Interface to Different Port-Size Devices	12-92
11.5.4	Interfacing to SDRAM	12-94
11.5.4.1	Basic SDRAM Capabilities of the Local Bus	12-94

Contents

Paragraph Number	Title	Page Number
11.5.4.2	Maximum Amount of SDRAM Supported	12-95
11.5.4.3	SDRAM Machine Limitations	12-96
11.5.4.3.1	Analysis of Maximum Row Number Due to Bank Select Mux	12-96
11.5.4.3.2	Bank Select Signals	12-96
11.5.4.3.3	SDRAM of 128 MBytes	12-96
11.5.4.3.4	256-MByte SDRAM	12-99
11.5.4.3.5	512-MByte SDRAM Memory	12-99
11.5.4.3.6	Power-Down Mode	12-100
11.5.4.3.7	Self- Refresh	12-101
11.5.4.3.8	SDRAM Timing	12-102
11.5.4.4	Parity Support for SDRAM	12-104
11.5.5	Interfacing to ZBT SRAM	12-104
11.5.6	Interfacing to DSP Host Ports	12-106
11.5.6.1	Interfacing to MSC8101 HDI16	12-106
11.5.6.1.1	HDI16 Peripherals	12-106
11.5.6.1.2	Physical Interconnections	12-108
11.5.6.1.3	Supporting Burst Transfers	12-110
11.5.6.1.4	Host 60x Bus:HDI16 Peripheral Interface Hardware Timings	12-110
11.5.6.2	Interfacing to MSC8102 DSI	12-111
11.5.6.2.1	DSI in Asynchronous SRAM-Like Mode	12-111
11.5.6.2.2	DSI in Synchronous Mode	12-114
11.5.6.2.3	Broadcast Accesses	12-121
11.5.6.3	Interfacing to EHPI from Texas Instruments TMS320Cxxxx DSPs	12-122
11.5.6.3.1	Expansion to Multiple DSPs	12-126

Chapter 12
Three-Speed Ethernet Controller

12.1	Introduction	13-1
12.1.1	Three-Speed Ethernet Controller Overview	13-7
12.2	Features	13-7
12.3	Modes of Operation	13-8
12.4	External Signals Description	13-9
12.4.1	Detailed Signal Descriptions	13-11
12.4.1.1	TSECn_COL—Collision (Input)	13-11
12.4.1.2	TSECn_CRS—Carrier Sense (Input)	13-11
12.4.1.3	TSECn_GTX_CLK—Gigabit Transmit Clock (Output)	13-11
12.4.1.4	EC_GTX_CLK125—Gigabit Transmit 125 MHz Source (Input)	13-12
12.4.1.5	EC_MDC—Management Data Clock (Output)	13-12
12.4.1.6	EC_MDIO—Management Data Input/Output, BIDI (Input/Output)	13-12
12.4.1.7	TSECn_RX_CLK—Receive Clock (Input)	13-12
12.4.1.8	TSECn_RX_DV—Receive Data Valid (Input)	13-12

Contents

Paragraph Number	Title	Page Number
12.4.1.9	TSECn_RXD[7:0]—Receive Data (Input)	13-12
12.4.1.10	TSECn_RX_ER—Receive Error (Input)	13-13
12.4.1.11	TSECn_TX_CLK—Transmit Clock (Input)	13-13
12.4.1.12	TSECn_TXD[7:0]—Transmit Data (Output)	13-13
12.4.1.13	TSECn_TX_EN—Transmit Data Valid (Output)	13-14
12.4.1.14	TSECn_TX_ER—Transmit Error (Output)	13-14
12.5	Memory Map/Register Definition	13-14
12.5.1	Top-Level Module Memory Map	13-15
12.5.2	Detailed Memory Map—Control/Status Registers	13-15
12.5.3	Memory-Mapped Register Descriptions	13-21
12.5.3.1	TSEC General Control and Status Registers	13-21
12.5.3.1.1	Interrupt Event Register (IEVENT)	13-21
12.5.3.1.2	Interrupt Mask Register (IMASK)	13-24
12.5.3.1.3	Error Disabled Register (EDIS)	13-25
12.5.3.1.4	Ethernet Control Register (ECNTRL)	13-26
12.5.3.1.5	Minimum Frame Length Register (MINFLR)	13-28
12.5.3.1.6	Pause Time Value Register (PTV)	13-28
12.5.3.1.7	DMA Control Register (DMACTRL)	13-29
12.5.3.1.8	TBI Physical Address Register (TBIPA)	13-31
12.5.3.2	TSEC FIFO Control and Status Registers	13-31
12.5.3.2.1	FIFO Transmit Threshold Register (FIFO_TX_THR)	13-32
12.5.3.2.2	FIFO Transmit Starve Register (FIFO_TX_STARVE)	13-33
12.5.3.2.3	FIFO Transmit Starve Shutoff Register (FIFO_TX_STARVE_SHUTOFF)	13-33
12.5.3.3	TSEC Transmit Control and Status Registers	13-34
12.5.3.3.1	Transmit Control Register (TCTRL)	13-34
12.5.3.3.2	Transmit Status Register (TSTAT)	13-35
12.5.3.3.3	TxBD Data Length Register (TBDLEN)	13-36
12.5.3.3.4	Current Transmit Buffer Descriptor Pointer (CTBPTR)	13-36
12.5.3.3.5	Transmit Descriptor Base Address (TBASE)	13-37
12.5.3.3.6	Out-of-Sequence TxBD Register (OSTBD)	13-38
12.5.3.3.7	Out-of-Sequence Tx Data Buffer Pointer Register (OSTBDP)	13-40
12.5.3.4	TSEC Receive Control and Status Registers	13-40
12.5.3.4.1	Receive Control Register (RCTRL)	13-41
12.5.3.4.2	Receive Status Register (RSTAT)	13-41
12.5.3.4.3	RxBD Data Length Register (RBDLEN)	13-42
12.5.3.4.4	Current Receive Buffer Descriptor Pointer (CRBPTR)	13-43
12.5.3.4.5	Maximum Receive Buffer Length Register (MRBLR)	13-43
12.5.3.4.6	Receive Buffer Descriptor Pointer (RBPTR)	13-44
12.5.3.4.7	Receive Descriptor Base Address Register (RBASE)	13-45
12.5.3.5	MAC Functionality	13-45

Contents

Paragraph Number	Title	Page Number
12.5.3.5.1	Configuring the MAC	13-46
12.5.3.5.2	Controlling CSMA/CD	13-46
12.5.3.5.3	Handling Packet Collisions	13-46
12.5.3.5.4	Controlling Packet Flow	13-47
12.5.3.5.5	Controlling PHY Links	13-48
12.5.3.6	MAC Registers	13-48
12.5.3.6.1	MAC Configuration 1 Register (MACCFG1)	13-48
12.5.3.6.2	MAC Configuration 2 Register (MACCFG2)	13-50
12.5.3.6.3	Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG)	13-51
12.5.3.6.4	Half-Duplex Register (HAFDUP)	13-52
12.5.3.6.5	Maximum Frame Length Register (MAXFRM)	13-54
12.5.3.6.6	MII Management Configuration Register (MIIMCFG)	13-54
12.5.3.6.7	MII Management Command (MIIMCOM)	13-55
12.5.3.6.8	MII Management Address Register (MIIMADD)	13-56
12.5.3.6.9	MII Management Control Register (MIIMCON)	13-57
12.5.3.6.10	MII Management Status Register (MIIMSTAT)	13-57
12.5.3.6.11	MII Management Indicator Register (MIIMIND)	13-58
12.5.3.6.12	Interface Status Register (IFSTAT)	13-58
12.5.3.6.13	Station Address Part 1 Register (MACSTNADDR1)	13-59
12.5.3.6.14	Station Address Part 2 Register (MACSTNADDR2)	13-60
12.5.3.7	MIB Registers	13-61
12.5.3.7.1	Transmit and Receive 64-Byte Frame Counter (TR64)	13-61
12.5.3.7.2	Transmit and Receive 65- to 127-Byte Frame Counter (TR127)	13-62
12.5.3.7.3	Transmit and Receive 128- to 255-Byte Frame Counter (TR255)	13-62
12.5.3.7.4	Transmit and Receive 256- to 511-Byte Frame Counter (TR511)	13-63
12.5.3.7.5	Transmit and Receive 512- to 1023-Byte Frame Counter (TR1K)	13-64
12.5.3.7.6	Transmit and Receive 1024- to 1518-Byte Frame Counter (TRMAX)	13-64
12.5.3.7.7	Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter (TRMGV)	13-65
12.5.3.7.8	Receive Byte Counter (RBYT)	13-66
12.5.3.7.9	Receive Packet Counter (RPKT)	13-66
12.5.3.7.10	Receive FCS Error Counter (RFCS)	13-67
12.5.3.7.11	Receive Multicast Packet Counter (RMCA)	13-68
12.5.3.7.12	Receive Broadcast Packet Counter (RBCA)	13-68
12.5.3.7.13	Receive Control Frame Packet Counter (RXCF)	13-69
12.5.3.7.14	Receive Pause Frame Packet Counter (RXPF)	13-70
12.5.3.7.15	Receive Unknown OPCode Packet Counter (RXUO)	13-70
12.5.3.7.16	Receive Alignment Error Counter (RALN)	13-71
12.5.3.7.17	Receive Frame Length Error Counter (RFLR)	13-72
12.5.3.7.18	Receive Code Error Counter (RCDE)	13-72
12.5.3.7.19	Receive Carrier Sense Error Counter (RCSE)	13-73

Contents

Paragraph Number	Title	Page Number
12.5.3.7.20	Receive Undersize Packet Counter (RUND)	13-74
12.5.3.7.21	Receive Oversize Packet Counter (ROVR)	13-74
12.5.3.7.22	Receive Fragments Counter (RFRG)	13-75
12.5.3.7.23	Receive Jabber Counter (RJBR)	13-76
12.5.3.7.24	Receive Dropped Packet Counter (RDRP)	13-76
12.5.3.7.25	Transmit Byte Counter (TBYT)	13-77
12.5.3.7.26	Transmit Packet Counter (TPKT)	13-78
12.5.3.7.27	Transmit Multicast Packet Counter (TMCA)	13-78
12.5.3.7.28	Transmit Broadcast Packet Counter (TBCA)	13-79
12.5.3.7.29	Transmit Pause Control Frame Counter (TXPF)	13-80
12.5.3.7.30	Transmit Deferral Packet Counter (TDFR)	13-80
12.5.3.7.31	Transmit Excessive Deferral Packet Counter (TEDF)	13-81
12.5.3.7.32	Transmit Single Collision Packet Counter (TSCL)	13-82
12.5.3.7.33	Transmit Multiple Collision Packet Counter (TMCL)	13-82
12.5.3.7.34	Transmit Late Collision Packet Counter (TLCL)	13-83
12.5.3.7.35	Transmit Excessive Collision Packet Counter (TXCL)	13-84
12.5.3.7.36	Transmit Total Collision Counter (TNCL)	13-84
12.5.3.7.37	Transmit Drop Frame Counter (TDRP)	13-85
12.5.3.7.38	Transmit Jabber Frame Counter (TJBR)	13-86
12.5.3.7.39	Transmit FCS Error Counter (TFCS)	13-86
12.5.3.7.40	Transmit Control Frame Counter (TXCF)	13-87
12.5.3.7.41	Transmit Oversize Frame Counter (TOVR)	13-88
12.5.3.7.42	Transmit Undersize Frame Counter (TUND)	13-88
12.5.3.7.43	Transmit Fragment Counter (TFRG)	13-89
12.5.3.7.44	Carry Register One (CAR1)	13-90
12.5.3.7.45	Carry Register Two (CAR2)	13-91
12.5.3.7.46	Carry Mask Register One (CAM1)	13-92
12.5.3.7.47	Carry Mask Register Two (CAM2)	13-93
12.5.3.8	Hash Function Registers	13-95
12.5.3.8.1	Individual Address Registers 0–7 (IADDR*n*)	13-95
12.5.3.8.2	Group Address Registers 0–7 (GADDR*n*)	13-95
12.5.3.8.3	Attribute Register (ATTR)	13-96
12.5.3.8.4	Attribute Extract Length and Extract Index Register (ATTRELI)	13-97
12.5.4	Ten-Bit Interface (TBI)	13-98
12.5.4.1	TBI MII Set Register Descriptions	13-98
12.5.4.2	Control Register (CR)	13-99
12.5.4.3	Status Register (SR)	13-100
12.5.4.4	AN Advertisement Register (ANA)	13-101
12.5.4.5	AN Link Partner Base Page Ability Register (ANLPBPA)	13-103
12.5.4.6	AN Expansion Register (ANEX)	13-104
12.5.4.7	AN Next Page Transmit Register (ANNPT)	13-105

Contents

Paragraph Number	Title	Page Number
12.5.4.8	AN Link Partner Ability Next Page Register (ANLPANP)	13-106
12.5.4.9	Extended Status Register (EXST)	13-107
12.5.4.10	Jitter Diagnostics Register (JD)	13-108
12.5.4.11	TBI Control Register (TBICON)	13-109
12.6	Functional Description	13-110
12.6.1	Connecting to Physical Interfaces	13-110
12.6.1.1	Media-Independent Interface (MII)	13-111
12.6.1.2	Gigabit Media-Independent Interface (GMII)	13-111
12.6.1.3	Reduced Gigabit Media-Independent Interface (RGMII)	13-112
12.6.1.4	Ten-Bit Interface (TBI)	13-113
12.6.1.5	Reduced Ten-Bit Interface (RTBI)	13-114
12.6.2	Gigabit Ethernet Channel Operation	13-118
12.6.2.1	Initialization Sequence	13-118
12.6.2.1.1	Hardware Controlled Initialization	13-118
12.6.2.1.2	User Initialization	13-118
12.6.2.2	Soft Reset and Reconfiguring Procedure	13-119
12.6.2.3	Gigabit Ethernet Frame Transmission	13-120
12.6.2.4	Gigabit Ethernet Frame Reception	13-122
12.6.2.5	RMON Support	13-123
12.6.2.6	Frame Recognition	13-123
12.6.2.6.1	Destination Address Recognition	13-123
12.6.2.6.2	Hash Table Algorithm	13-125
12.6.2.7	Flow Control	13-126
12.6.2.8	Interrupt Handling	13-126
12.6.2.9	Inter-Packet Gap Time	13-127
12.6.2.10	Internal and External Loop Back	13-128
12.6.2.11	Error-Handling Procedure	13-128
12.6.3	Buffer Descriptors	13-129
12.6.3.1	Data Buffer Descriptor	13-129
12.6.3.2	Transmit Data Buffer Descriptor (TxBD)	13-131
12.6.3.3	Receive Buffer Descriptor (RxBD)	13-133
12.7	Application Information	13-135
12.7.1	Interface Mode Configuration	13-135
12.7.1.1	MII Interface Mode	13-135
12.7.1.2	GMII Interface Mode	13-139
12.7.1.3	TBI Interface Mode	13-142
12.7.1.4	RGMII Interface Mode	13-145
12.7.1.5	RTBI Interface Mode	13-149

Contents

Paragraph Number	Title	Page Number
	Chapter 13 DMA Controller	
13.1	Introduction	14-1
13.1.1	Block Diagram	14-1
13.1.2	Overview	14-2
13.1.3	Features	14-2
13.1.4	Modes of Operation	14-2
13.2	External Signal Description	14-5
13.2.1	Signal Overview	14-5
13.2.2	Detailed Signal Descriptions	14-6
13.3	Memory Map/Register Definition	14-7
13.3.1	Module Memory Map	14-7
13.3.2	DMA Register Descriptions	14-10
13.3.2.1	Mode Registers (MR*n*)	14-10
13.3.2.2	Status Registers (SR*n*)	14-13
13.3.2.3	Current Link Descriptor Address Registers (CLNDAR*n*)	14-14
13.3.2.4	Source Attributes Registers (SATR*n*)	14-16
13.3.2.5	Source Address Registers (SAR*n*)	14-18
13.3.2.5.1	Source Address Registers For RapidIO Maintenance Reads (SARM*n*)	14-18
13.3.2.6	Destination Attributes Registers (DATR*n*)	14-19
13.3.2.7	Destination Address Registers (DAR*n*)	14-21
13.3.2.7.1	Destination Address Registers for RapidIO Maintenance Writes (DARM*n*)	14-22
13.3.2.8	Byte Count Registers (BCR*n*)	14-23
13.3.2.9	Next Link Descriptor Address Registers (NLNDAR*n*)	14-23
13.3.2.10	Current List Descriptor Address Registers (CLSDAR*n*)	14-24
13.3.2.11	Next List Descriptor Address Registers (NLSDAR*n*)	14-25
13.3.2.12	Source Stride Registers (SSR*n*)	14-26
13.3.2.13	Destination Stride Registers (DSR*n*)	14-26
13.3.2.14	DMA General Status Register (DGSR)	14-27
13.4	Functional Description	14-29
13.4.1	DMA Channel Operation	14-29
13.4.1.1	Basic DMA Mode Transfer	14-30
13.4.1.1.1	Basic Direct Mode	14-30
13.4.1.1.2	Basic Direct Single-Write Start Mode	14-31
13.4.1.1.3	Basic Chaining Mode	14-31
13.4.1.1.4	Basic Chaining Single-Write Start Mode	14-32
13.4.1.2	Extended DMA Mode Transfer	14-33
13.4.1.2.1	Extended Direct Mode	14-33
13.4.1.2.2	Extended Direct Single-Write Start Mode	14-33

Contents

Paragraph Number	Title	Page Number
13.4.1.2.3	Extended Chaining Mode	14-33
13.4.1.2.4	Extended Chaining Single-Write Start Mode	14-34
13.4.1.3	External Control Mode Transfer	14-34
13.4.1.4	Channel Continue Mode for Cascading Transfer Chains	14-35
13.4.1.4.1	Basic Mode	14-36
13.4.1.4.2	Extended Mode	14-36
13.4.1.5	Channel Abort	14-36
13.4.1.6	Bandwidth Control	14-37
13.4.1.7	Channel State	14-37
13.4.1.8	Illustration of Stride Size and Stride Distance	14-37
13.4.2	DMA Transfer Interfaces	14-38
13.4.3	DMA Errors	14-38
13.4.4	DMA Descriptors	14-38
13.4.5	Limitations and Restrictions	14-41
13.5	DMA System Considerations	14-42
13.5.1	Unusual DMA Scenarios	14-44
13.5.1.1	DMA to e500 Core	14-44
13.5.1.2	DMA to CPM	14-45
13.5.1.3	DMA to Ethernet	14-45
13.5.1.4	DMA to Configuration and Control Registers	14-45
13.5.1.5	DMA to I2C	14-46

Chapter 14
PCI Bus Interface

14.1	Introduction	15-1
14.1.1	Overview	15-2
14.1.1.1	MPC8560 as a PCI Initiator	15-4
14.1.1.2	MPC8560 as a PCI Target	15-4
14.1.2	Features	15-4
14.1.3	Modes of Operation	15-5
14.1.3.1	Host/Agent/Agent Lock Mode	15-5
14.1.3.1.1	PCI/PCI-X Signal Output Hold Timing	15-6
14.2	External Signal Descriptions	15-6
14.3	Memory Map/Register Definitions	15-13
14.3.1	PCI/PCI-X Memory Mapped Registers	15-14
14.3.1.1	PCI/X ATMU Outbound Registers	15-17
14.3.1.1.1	PCI/X Outbound Translation Address Registers (POTAR)	15-17
14.3.1.1.2	PCI/X Outbound Translation Extended Address Registers (POTEAR)	15-18
14.3.1.1.3	PCI/X Outbound Window Base Address Registers (POWBAR)	15-19
14.3.1.1.4	PCI/X Outbound Window Attributes Registers (POWAR)	15-19

Contents

Paragraph Number	Title	Page Number
14.3.1.2	PCI/X ATMU Inbound Registers	15-21
14.3.1.2.1	PCI/X Inbound Translation Address Registers (PITAR)	15-22
14.3.1.2.2	PCI/X Inbound Window Base Address Registers (PIWBAR)	15-23
14.3.1.2.3	PCI/X Inbound Window Base Extended Address Registers (PIWBEAR)	15-23
14.3.1.2.4	PCI/X Inbound Window Attributes Registers (PIWAR)	15-24
14.3.1.3	PCI/X Error Management Registers	15-26
14.3.1.3.1	PCI/X Error Detect Register	15-27
14.3.1.3.2	PCI/X Error Capture Disable Register	15-28
14.3.1.3.3	PCI/X Error Enable Register	15-29
14.3.1.3.4	PCI/X Error Attributes Capture Register	15-29
14.3.1.3.5	PCI/X Error Address Capture Register	15-31
14.3.1.3.6	PCI/X Error Extended Address Capture Register	15-31
14.3.1.3.7	PCI/X Error Data Low Capture Register	15-32
14.3.1.3.8	PCI/X Error Data High Capture Register	15-32
14.3.1.4	PCI/X Interrupt Acknowledge Register	15-33
14.3.2	PCI/PCI-X Configuration Header	15-33
14.3.2.1	Accessing PCI/X Configuration Header Registers	15-34
14.3.2.2	PCI Agent Mode Configuration Register Access	15-35
14.3.2.3	PCI Bus Command Register—Offset 0x04	15-35
14.3.2.4	PCI Bus Status Register—Offset 0x06	15-37
14.3.2.5	PCI Bus Programming Interface Register—Offset 0x09	15-38
14.3.2.6	PCI Bus Base Class Code Register—Offset 0x0B	15-38
14.3.2.7	PCI Bus Cache Line Size Register—Offset 0x0C	15-39
14.3.2.8	PCI Bus Latency Timer Register—0x0D	15-39
14.3.2.9	PCI Base Address Registers	15-40
14.3.2.10	PCI Bus Capabilities Pointer Register	15-42
14.3.2.11	PCI Bus Interrupt Line Register	15-43
14.3.2.12	PCI Bus Interrupt Pin Register	15-43
14.3.2.13	PCI Bus Maximum Grant (MAX GNT) Register	15-44
14.3.2.14	PCI Bus Maximum Latency (MAX LAT) Register	15-44
14.3.2.15	PCI Bus Function Register (PBFR)	15-45
14.3.2.16	PCI Bus Arbiter Configuration Register (PBACR)	15-45
14.3.2.17	PCI-X Next Capabilities ID Register—0x60	15-46
14.3.2.18	PCI-X Capability Pointer Register—0x61	15-47
14.3.2.19	PCI-X Command Register—0x62	15-47
14.3.2.20	PCI-X Status Register—0x64	15-48
14.4	Functional Description	15-49
14.4.1	PCI/PCI-X Bus Arbitration	15-49
14.4.1.1	PCI Bus Arbiter Operation	15-50
14.4.1.2	PCI Bus Parking	15-51

Contents

Paragraph Number	Title	Page Number
14.4.1.3	Broken Master Lock-Out	15-51
14.4.1.4	Power-Saving Modes and the PCI Arbiter	15-52
14.4.2	PCI Bus Protocol	15-52
14.4.2.1	Basic Transfer Control	15-52
14.4.2.2	PCI Bus Commands	15-53
14.4.2.3	Addressing	15-54
14.4.2.3.1	Memory Space Addressing	15-55
14.4.2.3.2	I/O Space Addressing	15-56
14.4.2.3.3	Configuration Space Addressing	15-56
14.4.2.4	Device Selection	15-56
14.4.2.5	Byte Alignment	15-57
14.4.2.6	Bus Driving and Turnaround	15-57
14.4.2.7	PCI Bus Transactions	15-57
14.4.2.7.1	PCI Read Transactions	15-58
14.4.2.7.2	PCI Write Transactions	15-59
14.4.2.8	Transaction Termination	15-60
14.4.2.8.1	Master-Initiated Termination	15-60
14.4.2.8.2	Target-Initiated Termination	15-61
14.4.2.9	Fast Back-to-Back Transactions	15-64
14.4.2.10	Dual Address Cycles	15-64
14.4.2.11	Configuration Cycles	15-66
14.4.2.11.1	PCI Configuration Space Header	15-66
14.4.2.11.2	Accessing the PCI Configuration Space	15-68
14.4.2.11.3	Type 0 Configuration Translation	15-70
14.4.2.11.4	Type 1 Configuration Translation	15-72
14.4.2.12	Other Bus Transactions	15-72
14.4.2.12.1	Interrupt-Acknowledge Transactions	15-73
14.4.2.12.2	Special-Cycle Transactions	15-73
14.4.2.13	PCI Error Functions	15-74
14.4.2.13.1	PCI Parity	15-74
14.4.2.13.2	Error Reporting	15-76
14.5	Initialization/Application Information	15-76
14.5.1	Power-On Reset Configuration Modes	15-76
14.5.1.1	Host Mode	15-77
14.5.1.2	Agent Mode	15-77
14.5.1.3	Agent Config Lock Mode	15-77
14.5.2	Nonposted Writes in PCI-X	15-77
14.5.3	Bridging between RapidIO and PCI/PCI-X	15-78

Contents

Paragraph Number	Title	Page Number
	Chapter 15 RapidIO Interface	
15.1	Introduction	16-1
15.1.1	RapidIO Terminology	16-2
15.1.2	Block Diagram	16-3
15.1.3	RapidIO Interface Features	16-4
15.1.3.1	Supported Features	16-5
15.1.3.2	Features Not Implemented	16-5
15.1.4	RapidIO Modes of Operation	16-5
15.1.4.1	Transmit Clock-Select Mode	16-5
15.1.4.2	CRC Checking Modes	16-5
15.1.4.3	Error Checking Disable Mode	16-6
15.1.4.4	Output Port Enable Mode	16-6
15.1.4.5	Output Port Driver Disable Mode	16-6
15.1.4.6	Accept-All Mode	16-6
15.1.4.7	Packet Response Time-Out Disable Mode	16-6
15.1.4.8	Link Response Time-Out Disable Mode	16-6
15.2	External Signal Description	16-6
15.2.1	Overview	16-7
15.2.2	Detailed Signal Descriptions	16-8
15.3	Memory Map/Register Definition	16-9
15.3.1	RapidIO Registers	16-9
15.3.2	Architectural Registers	16-14
15.3.2.1	Device Identity Capability Register (DIDCAR)	16-14
15.3.2.2	Device Information Capability Register (DICAR)	16-14
15.3.2.3	Assembly Identity Capability Register (AIDCAR)	16-15
15.3.2.4	Assembly Information Capability Register (AICAR)	16-16
15.3.2.5	Processing Element Features Capability Register (PEFCAR)	16-16
15.3.2.6	Switch Port Information Capability Register (SPICAR)	16-17
15.3.2.7	Source Operations Capability Register (SOCAR)	16-18
15.3.2.8	Destination Operations Capability Register (DOCAR)	16-19
15.3.2.9	Mailbox Command and Status Register (MSR)	16-21
15.3.2.10	Port-Write and Doorbell Command and Status Register (PWDCSR)	16-22
15.3.2.11	Processing Element Logic Layer Control Command and Status Register (PELLCCSR)	16-23
15.3.2.12	Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)	16-23
15.3.2.13	Base Device ID Command and Status Register (BDIDCSR)	16-24
15.3.2.14	Host Base Device ID Lock Command and Status Register (HBDIDLCSR)	16-25
15.3.2.15	Component Tag Command and Status Register (CTCSR)	16-25

Contents

Paragraph Number	Title	Page Number
15.3.2.16	8/16 LP-LVDS Port Maintenance Block Header 0 Command and Status Register (PMBH0CSR)	16-26
15.3.2.17	Port Link Time-Out Control Command and Status Register (PLTOCCSR)	16-27
15.3.2.18	Port Response Time-Out Control Command and Status Register (PRTOCCSR)	16-27
15.3.2.19	Port General Control Command and Status Register (PGCCSR)	16-28
15.3.2.20	Port Link Maintenance Request Command and Status Register (PLMREQCSR)	16-29
15.3.2.21	Port Link Maintenance Response Command and Status Register (PLMRESPCSR)	16-29
15.3.2.22	Port Local AckID Status Command and Status Register (PLASCSR)	16-30
15.3.2.23	Port Error and Status Command and Status Register (PESCSR)	16-31
15.3.2.24	Port Control Command and Status Register (PCCSR)	16-32
15.3.3	Implementation Registers	16-33
15.3.3.1	General Registers	16-33
15.3.3.1.1	Configuration Register (CR)	16-33
15.3.3.1.2	Port Configuration Register (PCR)	16-34
15.3.3.1.3	Port Error Injection Register (PEIR)	16-35
15.3.3.2	ATMU Registers	16-36
15.3.3.2.1	RapidIO Outbound Window Translation Address Registers 0–8 (ROWTAR*n*)	16-36
15.3.3.2.2	RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR*n*)	16-38
15.3.3.2.3	RapidIO Outbound Window Attributes Registers 0–8 (ROWAR*n*)	16-39
15.3.3.2.4	RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR*n*)	16-41
15.3.3.2.5	RapidIO Inbound Window Base Address Registers 1–4 (RIWBAR*n*)	16-42
15.3.3.2.6	RapidIO Inbound Window Attributes Registers 0–4 (RIWAR*n*)	16-42
15.3.3.3	Error Management Registers	16-44
15.3.3.3.1	Port Notification/Fatal Error Detect Register (PNFEDR)	16-44
15.3.3.3.2	Port Notification/Fatal Error Disable Register (PNFEDiR)	16-47
15.3.3.3.3	Port Notification/Fatal Error Interrupt Enable Register (PNFEIER)	16-49
15.3.3.3.4	Port Error Control Register (PECR)	16-50
15.3.3.3.5	Port Error Packet/Control Symbol Register 0 (PEPCSR0)	16-51
15.3.3.3.6	Port Error Packet Register 1 (PEPR1)	16-52
15.3.3.3.7	RapidIO Packet Error Capture Register 1, Type 1 (PECR1T1)	16-52
15.3.3.3.8	RapidIO Packet Error Capture Register 1, Type 2 (PECR1T2)	16-53
15.3.3.3.9	RapidIO Packet Error Capture Register 1, Type 5 (PECR1T5)	16-53
15.3.3.3.10	RapidIO Packet Error Capture Register 1, Type 6 (PECR1T6)	16-54

Contents

Paragraph Number	Title	Page Number
15.3.3.3.11	RapidIO Packet Error Capture Register 1, Type 8 Request (PECR1T8RQ)	16-55
15.3.3.3.12	RapidIO Packet Error Capture Register 1, Type 8 Response (PECR1T8R)	16-55
15.3.3.3.13	RapidIO Packet Error Capture Register 1, Type 10 (PECR1T10)	16-56
15.3.3.3.14	RapidIO Packet Error Capture Register 1, Type 11 (PECR1T11)	16-57
15.3.3.3.15	RapidIO Packet Error Capture Register 1, Type 13 (PECR1T13)	16-57
15.3.3.3.16	RapidIO Port Error Packet Register 2 (PEPR2)	16-58
15.3.3.3.17	RapidIO Packet Error Capture Register 2, Types 1, 2, and 5 (PECR2T125)	16-58
15.3.3.3.18	RapidIO Packet Error Capture Register 2, Type 6 (PECR2T6)	16-59
15.3.3.3.19	RapidIO Packet Error Capture Register 2, Type 8 (PECR2T8)	16-60
15.3.3.3.20	RapidIO Packet Error Capture Register 2, Type 8 Response (PECR2T8R)	16-60
15.3.3.3.21	RapidIO Packet Error Capture Register 2, Type 10 (PECR2T10)	16-61
15.3.3.3.22	RapidIO Packet Error Capture Register 2, Types 11 and 13 (PECR2T11)	16-62
15.3.3.3.23	Port Recoverable Error Detect Register (PREDR)	16-62
15.3.3.3.24	Port Error Recovery Threshold Register (PERTR)	16-65
15.3.3.3.25	Port Retry Threshold Register (PRTR)	16-66
15.3.4	RapidIO Message Unit Registers	16-67
15.3.4.1	RapidIO Outbound Message Registers	16-67
15.3.4.1.1	Outbound Mode Register (OMR)	16-67
15.3.4.1.2	Outbound Status Register (OSR)	16-69
15.3.4.1.3	Outbound Descriptor Queue Dequeue Pointer Address Register (ODQDPAR)	16-70
15.3.4.1.4	Outbound Unit Source Address Register (OSAR)	16-71
15.3.4.1.5	Outbound Destination Port Register (ODPR)	16-72
15.3.4.1.6	Outbound Destination Attributes Register (ODATR)	16-72
15.3.4.1.7	Outbound Double-Word Count Register (ODCR)	16-73
15.3.4.1.8	Outbound Descriptor Queue Enqueue Pointer Address Register (ODQEPAR)	16-74
15.3.4.2	RapidIO Inbound Message Registers	16-75
15.3.4.2.1	Inbound Mode Register (IMR)	16-75
15.3.4.2.2	Inbound Status Register (ISR)	16-77
15.3.4.2.3	Inbound Frame Queue Dequeue Pointer Address Register (IFQDPAR)	16-79
15.3.4.2.4	Inbound Frame Queue Enqueue Pointer Address Register (IFQEPAR)	16-79
15.3.4.3	RapidIO Doorbell Registers	16-80
15.3.4.3.1	Doorbell Mode Register (DMR)	16-80

Contents

Paragraph Number	Title	Page Number
15.3.4.3.2	Doorbell Status Register (DSR)	16-82
15.3.4.3.3	Doorbell Queue/Dequeue Pointer Address Registers (DQDPAR)	16-83
15.3.4.3.4	Doorbell Queue/Enqueue Pointer Address Registers (DQEPAR)	16-84
15.3.4.4	RapidIO Port-Write Registers	16-85
15.3.4.4.1	Port-Write Mode Register (PWMR)	16-85
15.3.4.4.2	Port-Write Status Register (PWSR)	16-86
15.3.4.4.3	Port-Write Queue Base Address Register (PWQBAR)	16-87
15.4	Functional Description	16-88
15.4.1	RapidIO Transaction, Packet, and Control Symbol Summary	16-88
15.5	RapidIO Functionality	16-94
15.5.1	General Device Functionality Checklists	16-94
15.5.1.1	General Specification Functionality List	16-94
15.5.1.2	8/16 LP-LVDS Layer Functionality Lists	16-95
15.5.1.3	Basic Functionality List	16-95
15.5.1.4	General Device Link Maintenance List	16-97
15.5.1.5	General Device Packet Transmission List	16-97
15.5.1.6	General Device Packet Reception List	16-98
15.5.1.7	General Device Recoverable Errors List	16-100
15.5.1.8	General Device Nonrecoverable Errors List	16-102
15.5.2	Common Transport Layer Functionality Lists	16-103
15.5.2.1	Common Transport Layer Basic Functionality List	16-103
15.5.2.2	Common Transport Layer Packet Transmission List	16-103
15.5.2.3	Common Transport Layer Packet Reception List	16-103
15.5.2.4	Common Transport Layer Detectable Errors List	16-104
15.5.3	Logical Layer Functionality Lists	16-104
15.5.3.1	Logical Layer Basic Functionality List	16-104
15.5.3.2	Logical Layer Target Transaction Support List	16-105
15.5.3.3	Logical Layer Source Transaction Support List	16-106
15.5.3.4	Logical Layer Detectable Errors List	16-106
15.5.3.5	Logical Layer Extended Functionality Checklist	16-106
15.6	RapidIO Errors	16-106
15.7	ATMU (Address Translation and Mapping Unit)	16-112
15.7.1	Outbound ATMU Translation	16-113
15.7.1.1	Outbound ATMU Bypass Mode	16-113
15.7.1.2	Outbound Special Transactions and Requirements	16-113
15.7.2	Inbound ATMU Translation	16-114
15.7.2.1	Inbound ATMU LCSBA1CSR Window	16-114
15.7.2.2	Inbound ATMU Bypass Mode	16-114
15.7.2.3	Inbound Special Transactions and Requirements	16-115
15.8	RapidIO Message Unit	16-116
15.8.1	Overview	16-116

Contents

Paragraph Number	Title	Page Number
15.8.2	Features	16-116
15.8.3	Modes of Operation	16-117
15.8.4	RapidIO Messaging Description	16-117
15.8.4.1	Data Message Controller	16-117
15.8.4.2	Outbox Controller Operation	16-118
15.8.4.2.1	Direct Mode Operation	16-118
15.8.4.2.2	Chaining Mode Operation	16-118
15.8.4.2.3	Switching Between Direct and Chaining Modes	16-122
15.8.4.2.4	Descriptor Format	16-122
15.8.4.2.5	Outbox Controller Interrupts	16-123
15.8.4.2.6	Special Error Case Condition	16-123
15.8.4.3	Inbox Controller Operation	16-123
15.8.4.3.1	Retry Response Conditions	16-125
15.8.4.3.2	Error Response Conditions	16-125
15.8.4.3.3	Inbox Controller Interrupts	16-125
15.8.4.3.4	Data Message Controller Limitations and Restrictions	16-125
15.8.4.4	Doorbell Message Controller	16-125
15.8.4.4.1	Inbound Doorbell Reception	16-126
15.8.4.4.2	Doorbell Queue Entry Format	16-127
15.8.4.4.3	Retry Response Conditions	16-127
15.8.4.4.4	Error Response Conditions	16-127
15.8.4.4.5	Doorbell Controller Interrupts	16-127
15.8.4.5	Port-Write Controller Structure	16-128
15.9	Initialization and Application Information	16-129

Part IV - Global Functions and Debug

Chapter 16
Global Utilities

16.1	Introduction	17-1
16.1.1	Overview	17-1
16.1.2	Global Utilities Features	17-1
16.1.2.1	Power Management and Device Disables	17-1
16.1.2.2	Software Access to Current POR Configuration Settings	17-1
16.1.2.3	General-Purpose I/O	17-2
16.1.2.4	Interrupt and Local Bus Signal Multiplexing	17-2
16.1.2.5	Clock Control	17-2
16.2	External Signal Description	17-2
16.2.1	Signals Overview	17-2
16.2.2	Detailed Signal Descriptions	17-3
16.3	Memory Map/Register Definition	17-4

Contents

Paragraph Number	Title	Page Number
16.3.1	Register Descriptions	17-5
16.3.1.1	POR PLL Status Register (PORPLLSR)	17-5
16.3.1.2	POR Boot Mode Status Register (PORBMSR)	17-6
16.3.1.3	POR I/O Impedance Status and Control Register (PORIMPSCR)	17-8
16.3.1.4	POR Device Status Register (PORDEVSR)	17-9
16.3.1.5	POR Debug Mode Status Register (PORDBGMSR)	17-10
16.3.1.6	General-Purpose POR Configuration Register (GPPORCR)	17-11
16.3.1.7	General-Purpose I/O Control Register (GPIOCR)	17-11
16.3.1.8	General-Purpose Output Data Register (GPOUTDR)	17-12
16.3.1.9	General-Purpose Input Data Register (GPINDR)	17-13
16.3.1.10	Alternate Function Signal Multiplex Control Register (PMUXCR)	17-14
16.3.1.11	Device Disable Register (DEVDISR)	17-15
16.3.1.12	Power Management Control and Status Register (POWMGTCSR)	17-17
16.3.1.13	Machine Check Summary Register (MCPSUMR)	17-19
16.3.1.14	Processor Version Register (PVR)	17-20
16.3.1.15	System Version Register (SVR)	17-20
16.3.1.16	Clock Out Control Register (CLKOCR)	17-21
16.3.1.17	DDR DLL Control Register (DDRDLLCR)	17-21
16.3.1.18	Local Bus DLL Control Register (LBDLLCR)	17-22
16.4	Functional Description	17-23
16.4.1	Power Management	17-23
16.4.1.1	Dynamic Power Management	17-24
16.4.1.2	Shutting Down Unused Blocks	17-24
16.4.1.3	Software-Controlled Power-Down States	17-24
16.4.1.4	Doze Mode	17-24
16.4.1.5	Nap Mode	17-25
16.4.1.6	Sleep Mode	17-25
16.4.1.7	Power Management Control Bits	17-25
16.4.1.8	Interrupts and Power Management	17-26
16.4.1.9	Snooping in the Power-Down State	17-26
16.4.1.10	Software Considerations for Power Management	17-26
16.4.2	General-Purpose I/O Signals	17-27
16.4.3	Interrupt and Local Bus Signal Multiplexing	17-27

Chapter 17
Performance Monitor

17.1	Introduction	18-1
17.1.1	Overview	18-1
17.1.2	Features	18-2
17.2	External Signals	18-3
17.3	Memory Map and Register Definition	18-3

Contents

Paragraph Number	Title	Page Number
17.3.1	Register Summary	18-3
17.3.2	Control Registers	18-4
17.3.2.1	Performance Monitor Global Control Register (PMGC0)	18-5
17.3.2.2	Performance Monitor Local Control Registers (PMLCAn and PMLCB*n*)	18-5
17.3.3	Counter Registers	18-10
17.3.3.1	Performance Monitor Counters (PMC0, PMC1, PMC2, PMC3, PMC4, PMC5, PMC6, PMC7, PMC8)	18-10
17.4	Functional Description	18-12
17.4.1	Performance Monitor Interrupt	18-12
17.4.2	Event Counting	18-12
17.4.3	Threshold Events	18-13
17.4.4	Chaining	18-14
17.4.5	Triggering	18-14
17.4.6	Burstiness Counting	18-15
17.4.7	Performance Monitor Events	18-17
17.4.8	Performance Monitor Examples	18-38

Chapter 18
Debug Features and Watchpoint Facility

18.1	Introduction	19-1
18.1.1	Overview	19-2
18.1.2	Features	19-3
18.1.3	Modes of Operation	19-3
18.1.3.1	Local Bus Interface Debug Modes of Operation	19-4
18.1.3.2	DDR SDRAM Interface Debug Modes of Operation	19-4
18.1.3.3	PCI/PCI-X Interface Debug Modes of Operation	19-4
18.1.3.4	Watchpoint Monitor Modes of Operation	19-4
18.1.3.5	Trace Buffer Modes of Operation	19-5
18.2	External Signal Description	19-5
18.2.1	Overview	19-5
18.2.2	Detailed Signal Descriptions	19-6
18.2.2.1	Memory Data Valid (MDVAL)—Output	19-6
18.2.2.2	Memory ECC (MECC[0:7])—Output	19-6
18.2.2.3	Memory Source ID (MSRCID[0:4])—Output	19-7
18.2.2.4	Watchpoint Trigger Signals	19-7
18.2.2.4.1	Trigger In (TRIG_IN)—Input	19-7
18.2.2.4.2	Trigger Out (TRIG_OUT)—Output	19-8
18.2.2.5	PCI Address (PCI_AD[63:59])—Output	19-8
18.2.3	Test and Configuration Signals	19-8
18.2.3.1	JTAG Test Clock (TCK)—Input	19-8
18.2.3.2	JTAG Test Data Input (TDI)—Input	19-9

Contents

Paragraph Number	Title	Page Number
18.2.3.3	JTAG Test Data Output (TDO)—Output	19-9
18.2.3.4	JTAG Test Mode Select (TMS)—Input	19-9
18.2.3.5	JTAG Test Reset ($\overline{\text{TRST}}$)—Input	19-9
18.2.3.6	LSSD_MODE—Input	19-10
18.2.3.7	L1_TSTCLK—Input	19-10
18.2.3.8	L2_TSTCLK—Input	19-10
18.2.3.9	TEST_SEL—Input	19-10
18.3	Memory Map/Register Definition	19-10
18.3.1	Watchpoint Monitor Register Descriptions	19-11
18.3.1.1	Watchpoint Monitor Control Register 0 (WMCR0)	19-11
18.3.1.2	Watchpoint Monitor Control Register 1 (WMCR1)	19-12
18.3.1.3	Watchpoint Monitor Address Register (WMAR)	19-13
18.3.1.4	Watchpoint Monitor Address Mask Register (WMAMR)	19-14
18.3.1.5	Watchpoint Monitor Transaction Mask Register (WMTMR)	19-15
18.3.1.6	Watchpoint Monitor Status Register (WMSR)	19-17
18.3.2	Trace Buffer Register Descriptions	19-18
18.3.2.1	Trace Buffer Control Register (TBCR0)	19-18
18.3.2.2	Trace Buffer Control Register 1 (TBCR1)	19-20
18.3.2.3	Trace Buffer Address Register (TBAR)	19-20
18.3.2.4	Trace Buffer Address Mask Register (TBAMR)	19-21
18.3.2.5	Trace Buffer Transaction Mask Register (TBTMR)	19-22
18.3.2.6	Trace Buffer Status Register (TBSR)	19-22
18.3.2.7	Trace Buffer Access Control Register (TBACR)	19-23
18.3.2.8	Trace Buffer Access Data High Register (TBADHR)	19-24
18.3.2.9	Trace Buffer Access Data Register (TBADR)	19-25
18.3.3	Context ID Registers	19-25
18.3.3.1	Programmed Context ID Register (PCIDR)	19-25
18.3.3.2	Current Context ID Register (CCIDR)	19-26
18.3.4	Trigger Out Function	19-27
18.3.4.1	Trigger Out Source Register (TOSR)	19-27
18.4	Functional Description	19-27
18.4.1	PCI/PCI-X Interface Debug	19-29
18.4.2	DDR SDRAM Interface Debug	19-29
18.4.2.1	Debug Information on Debug Pins	19-29
18.4.2.2	Debug Information on ECC Pins	19-30
18.4.3	Local Bus Interface Debug	19-30
18.4.4	Watchpoint Monitor	19-30
18.4.4.1	Watchpoint Monitor Performance Monitor Events	19-30
18.4.5	Trace Buffer	19-31
18.4.5.1	Traced Data (TBCR[TRSEL])	19-31
18.5	Initialization	19-33

Contents

Paragraph Number	Title	Page Number

Part V - CPM Features

Chapter 19
Communications Processor Module Overview

19.1	Features	20-1
19.1.1	CPM Memory Map	20-3
19.2	MPC8560 Serial Configurations	20-19
19.3	Communications Processor (CP)	20-20
19.3.1	Features	20-20
19.3.2	CP Block Diagram	20-21
19.3.3	e500 Core Interface	20-21
19.3.4	Peripheral Interface	20-22
19.3.5	Execution from RAM	20-24
19.3.6	RISC Controller Configuration Register (RCCR)	20-24
19.3.7	RISC Time-Stamp Control Register (RTSCR)	20-25
19.3.8	RISC Time-Stamp Register (RTSR)	20-26
19.3.9	RISC Microcode Revision Number	20-27
19.4	Command Set	20-27
19.4.1	CP Command Register (CPCR)	20-27
19.4.1.1	CP Commands	20-29
19.4.2	Command Register Example	20-31
19.4.3	Command Execution Latency	20-31
19.5	Internal RAM	20-31
19.5.1	Buffer Descriptors (BDs)	20-35
19.5.2	Parameter RAM	20-35
19.6	RISC Timer Tables	20-36
19.6.1	RISC Timer Table Parameter RAM	20-37
19.6.2	RISC Timer Command Register (TM_CMD)	20-38
19.6.3	RISC Timer Table Entries	20-39
19.6.4	RISC Timer Event Register (RTER)/Mask Register (RTMR)	20-39
19.6.5	set timer Command	20-40
19.6.6	RISC Timer Initialization Sequence	20-40
19.6.7	RISC Timer Initialization Example	20-40
19.6.8	RISC Timer Interrupt Handling	20-41
19.6.9	RISC Timer Table Scan Algorithm	20-41
19.6.10	Using the RISC Timers to Track CP Loading	20-42

Contents

Paragraph Number	Title	Page Number
	Chapter 20 CPM Interrupt Controller	
20.1	Interrupt Configuration	21-1
20.2	CPM Interrupt Source Priorities	21-2
20.2.1	SCC, FCC, and MCC Relative Priority	21-5
20.2.2	Highest Priority Interrupt	21-5
20.3	Masking Interrupt Sources	21-5
20.4	CPM Interrupt Vector Generation and Calculation	21-6
20.4.1	Port C External Interrupts	21-8
20.5	CPM Interrupt Programming Model	21-8
20.5.1	Interrupt Controller Registers	21-9
20.5.1.1	CPM Interrupt Configuration Register (SICR)	21-9
20.5.1.2	CPM Interrupt Priority Registers (SCPRR_H and SCPRR_L)	21-9
20.5.1.3	CPM Interrupt Pending Registers (SIPNR_H and SIPNR_L)	21-11
20.5.1.4	CPM Interrupt Mask Registers (SIMR_H and SIMR_L)	21-13
20.5.1.5	CPM Interrupt Vector Register (SIVEC)	21-14
20.5.1.6	CPM External Interrupt Control Register (SIEXR)	21-15
	Chapter 21 Serial Interface with Time-Slot Assigner	
21.1	Features	22-3
21.2	Overview	22-4
21.3	Enabling Connections to TSA	22-7
21.4	Serial Interface RAM	22-8
21.4.1	One Multiplexed Channel with Static Frames	22-9
21.4.2	One Multiplexed Channel with Dynamic Frames	22-9
21.4.3	Programming SIx RAM Entries	22-10
21.4.4	SIx RAM Programming Example	22-13
21.4.5	Static and Dynamic Routing	22-15
21.5	Serial Interface Registers	22-18
21.5.1	SI Global Mode Registers (SIxGMR)	22-18
21.5.2	SI Mode Registers (SIxMR)	22-18
21.5.3	SIx RAM Shadow Address Registers (SIxRSR)	22-24
21.5.4	SI Command Register (SIxCMDR)	22-25
21.5.5	SI Status Registers (SIxSTR)	22-26
21.6	Serial Interface IDL Interface Support	22-26
21.6.1	IDL Interface Programming	22-27
21.7	Serial Interface GCI Support	22-29
21.7.1	Serial Interface GCI Programming	22-30

Contents

Paragraph Number	Title	Page Number
21.7.1.1	Normal Mode GCI Programming	22-30
21.7.1.2	SCIT Programming	22-31

Chapter 22
CPM Multiplexing

22.1	Features	23-2
22.2	Enabling Connections to TSA or NMSI	23-3
22.3	NMSI Configuration	23-4
22.4	CMX Registers	23-7
22.4.1	CMX UTOPIA Address Register (CMXUAR)	23-7
22.4.2	CMX SI1 Clock Route Register (CMXSI1CR)	23-11
22.4.3	CMX SI2 Clock Route Register (CMXSI2CR)	23-11
22.4.4	CMX FCC Clock Route Register (CMXFCR)	23-12
22.4.5	CMX SCC Clock Route Register (CMXSCR)	23-15

Chapter 23
Baud-Rate Generators (BRGs)

23.1	System Clock Control Register (SCCR)	24-2
23.2	BRG Configuration Registers 1–8 (BRGCx)	24-3
23.3	Autobaud Operation on a UART	24-5
23.4	UART Baud Rate Examples	24-6

Chapter 24
CPM Timers

24.1	Features	25-2
24.2	General-Purpose Timer Units	25-2
24.2.1	Cascaded Mode	25-3
24.2.2	Timer Global Configuration Registers (TGCR1 and TGCR2)	25-4
24.2.3	Timer Mode Registers (TMR1–TMR4)	25-6
24.2.4	Timer Reference Registers (TRR1–TRR4)	25-7
24.2.5	Timer Capture Registers (TCR1–TCR4)	25-8
24.2.6	Timer Counters (TCN1–TCN4)	25-8
24.2.7	Timer Event Registers (TER1–TER4)	25-8

Chapter 25
SDMA Channels

25.1	SDMA Registers	26-2
25.1.1	SDMA Status Register (SDSR)	26-2

Contents

Paragraph Number	Title	Page Number
25.1.2	SDMA Mask Register (SDMR)	26-3
25.1.3	Address Error Registers (SMAER and LMAER)	26-3
25.1.4	Status Error Registers (SMSER and LMSER)	26-3
25.1.5	Event Registers (SMEVR and LMEVR)	26-4

Chapter 26
Serial Communications Controllers (SCCs)

26.1	Features	27-2
26.2	General SCC Mode Registers (GSMR1–GSMR4)	27-3
26.2.1	Protocol-Specific Mode Register (PSMR)	27-9
26.2.2	Data Synchronization Register (DSR)	27-9
26.2.3	Transmit-on-Demand Register (TODR)	27-10
26.3	SCC Buffer Descriptors (BDs)	27-11
26.4	SCC Parameter RAM	27-13
26.4.1	SCC Base Addresses	27-15
26.4.2	Function Code Registers (RFCR and TFCR)	27-15
26.4.3	Handling SCC Interrupts	27-16
26.4.4	Initializing the SCCs	27-17
26.4.5	Controlling SCC Timing with RTS, CTS, and CD	27-18
26.4.5.1	Synchronous Protocols	27-18
26.4.5.2	Asynchronous Protocols	27-21
26.4.6	Digital Phase-Locked Loop (DPLL) Operation	27-22
26.4.6.1	Encoding Data with a DPLL	27-24
26.4.7	Clock Glitch Detection	27-26
26.4.8	Reconfiguring the SCCs	27-26
26.4.8.1	General Reconfiguration Sequence for an SCC Transmitter	27-26
26.4.8.2	Reset Sequence for an SCC Transmitter	27-27
26.4.8.3	General Reconfiguration Sequence for an SCC Receiver	27-27
26.4.8.4	Reset Sequence for an SCC Receiver	27-27
26.4.8.5	Switching Protocols	27-27
26.4.9	Saving Power	27-28

Chapter 27
SCC UART Mode

27.1	Features	28-2
27.2	Normal Asynchronous Mode	28-3
27.3	Synchronous Mode	28-3
27.4	SCC UART Parameter RAM	28-4
27.5	Data-Handling Methods: Character- or Message-Based	28-5
27.6	Error and Status Reporting	28-6

Contents

Paragraph Number	Title	Page Number
27.7	SCC UART Commands	28-6
27.8	Multidrop Systems and Address Recognition	28-7
27.9	Receiving Control Characters	28-8
27.10	Hunt Mode (Receiver)	28-10
27.11	Inserting Control Characters into the Transmit Data Stream	28-10
27.12	Sending a Break (Transmitter)	28-11
27.13	Sending a Preamble (Transmitter)	28-11
27.14	Fractional Stop Bits (Transmitter)	28-11
27.15	Handling Errors in the SCC UART Controller	28-12
27.16	UART Mode Register (PSMR)	28-14
27.17	SCC UART Receive Buffer Descriptor (RxBD)	28-15
27.18	SCC UART Transmit Buffer Descriptor (TxBD)	28-19
27.19	SCC UART Event Register (SCCE) and Mask Register (SCCM)	28-20
27.20	SCC UART Status Register (SCCS)	28-22
27.21	SCC UART Programming Example	28-23
27.22	S-Records Loader Application	28-24

Chapter 28
SCC HDLC Mode

28.1	SCC HDLC Features	29-2
28.2	SCC HDLC Channel Frame Transmission	29-2
28.3	SCC HDLC Channel Frame Reception	29-3
28.4	SCC HDLC Parameter RAM	29-3
28.5	Programming the SCC in HDLC Mode	29-5
28.6	SCC HDLC Commands	29-5
28.7	Handling Errors in the SCC HDLC Controller	29-6
28.8	HDLC Mode Register (PSMR)	29-7
28.9	SCC HDLC Receive Buffer Descriptor (RxBD)	29-8
28.10	SCC HDLC Transmit Buffer Descriptor (TxBD)	29-12
28.11	HDLC Event Register (SCCE)/HDLC Mask Register (SCCM)	29-13
28.12	SCC HDLC Status Register (SCCS)	29-15
28.13	SCC HDLC Programming Examples	29-15
28.13.1	SCC HDLC Programming Example #1	29-16
28.13.2	SCC HDLC Programming Example #2	29-17
28.14	HDLC Bus Mode with Collision Detection	29-18
28.14.1	HDLC Bus Features	29-20
28.14.2	Accessing the HDLC Bus	29-20
28.14.3	Increasing Performance	29-21
28.14.4	Delayed RTS Mode	29-22
28.14.5	Using the Time-Slot Assigner (TSA)	29-23
28.14.6	HDLC Bus Protocol Programming	29-24

Contents

Paragraph Number	Title	Page Number
28.14.6.1	Programming GSMR and PSMR for the HDLC Bus Protocol	29-24
28.14.6.2	HDLC Bus Controller Programming Example	29-24

Chapter 29
SCC BISYNC Mode

29.1	Features	30-2
29.2	SCC BISYNC Channel Frame Transmission	30-2
29.3	SCC BISYNC Channel Frame Reception	30-3
29.4	SCC BISYNC Parameter RAM	30-4
29.5	SCC BISYNC Commands	30-5
29.6	SCC BISYNC Control Character Recognition	30-6
29.7	BISYNC SYNC Register (BSYNC)	30-7
29.8	SCC BISYNC DLE Register (BDLE)	30-8
29.9	Sending and Receiving the Synchronization Sequence	30-9
29.10	Handling Errors in the SCC BISYNC	30-10
29.11	BISYNC Mode Register (PSMR)	30-10
29.12	**SCC** BISYNC Receive BD (RxBD)	30-12
29.13	SCC BISYNC Transmit BD (TxBD)	30-14
29.14	BISYNC Event Register (SCCE)/BISYNC Mask Register (SCCM)	30-15
29.15	SCC Status Registers (SCCS)	30-16
29.16	Programming the SCC BISYNC Controller	30-17
29.17	SCC BISYNC Programming Example	30-18

Chapter 30
SCC Transparent Mode

30.1	Features	31-1
30.2	SCC Transparent Channel Frame Transmission Process	31-2
30.3	SCC Transparent Channel Frame Reception Process	31-3
30.4	Achieving Synchronization in Transparent Mode	31-3
30.4.1	Synchronization in NMSI Mode	31-3
30.4.1.1	In-Line Synchronization Pattern	31-3
30.4.1.2	External Synchronization Signals	31-4
30.4.1.2.1	External Synchronization Example	31-4
30.4.1.3	Transparent Mode without Explicit Synchronization	31-5
30.4.2	Synchronization and the TSA	31-6
30.4.2.1	Inline Synchronization Pattern	31-6
30.4.2.2	Inherent Synchronization	31-6
30.4.3	End of Frame Detection	31-6
30.5	CRC Calculation in Transparent Mode	31-6
30.6	SCC Transparent Parameter RAM	31-7

Contents

Paragraph Number	Title	Page Number
30.7	SCC Transparent Commands	31-7
30.8	Handling Errors in the Transparent Controller	31-8
30.9	Transparent Mode and the PSMR	31-9
30.10	SCC Transparent Receive Buffer Descriptor (RxBD)	31-9
30.11	SCC Transparent Transmit Buffer Descriptor (TxBD)	31-11
30.12	SCC Transparent Event Register (SCCE)/Mask Register (SCCM)	31-12
30.13	SCC Status Register in Transparent Mode (SCCS)	31-13
30.14	SCC2 Transparent Programming Example	31-14
	Chapter 31 SCC AppleTalk Mode	
31.1	Operating the LocalTalk Bus	32-1
31.2	Features	32-2
31.3	Connecting to AppleTalk	32-3
31.4	Programming the SCC in AppleTalk Mode	32-3
31.4.1	Programming the GSMR	32-3
31.4.2	Programming the PSMR	32-4
31.4.3	Programming the TODR	32-4
31.4.4	SCC AppleTalk Programming Example	32-4
	Chapter 32 Multi-Channel Controllers (MCCs)	
32.1	Features	33-1
32.2	SS7 Controller	33-2
32.2.1	SS7 Controller Features	33-2
32.3	MCC Data Structure Organization	33-3
32.4	Global MCC Parameters	33-5
32.5	Channel Extra Parameters	33-6
32.6	Super-Channel Table	33-7
32.7	Channel-Specific HDLC Parameters	33-10
32.7.1	Internal Transmitter State (TSTATE)	33-11
32.7.2	Interrupt Mask (INTMSK)	33-11
32.7.3	Channel Mode Register (CHAMR)	33-12
32.7.4	Internal Receiver State (RSTATE)	33-13
32.8	Channel-Specific Transparent Parameters	33-14
32.8.1	Channel Mode Register (CHAMR)—Transparent Mode	33-15
32.9	Channel-Specific SS7 Parameters	33-17
32.9.1	Extended Channel Mode Register (ECHAMR)	33-19
32.9.2	Signal Unit Error Monitor (SUERM)	33-21
32.9.3	SUERM in Japanese SS7	33-22

Contents

Paragraph Number	Title	Page Number
32.9.4	SS7 Configuration Register (SS7_OPT)	33-22
32.9.4.1	AERM Implementation	33-23
32.9.4.2	Japanese SS7	33-23
32.9.4.3	Disabling SUERM	33-24
32.9.4.4	SU Filtering	33-24
32.9.4.4.1	Comparison Mask	33-24
32.9.4.4.2	Comparison State Machine	33-25
32.9.4.4.3	Filtering Limitations	33-25
32.9.4.4.4	Resetting the SU Filtering Mechanism	33-26
32.9.4.5	Octet Counting Mode	33-26
32.10	MCC Configuration Registers (MCCFx)	33-26
32.11	MCC Commands	33-27
32.12	MCC Exceptions	33-28
32.12.1	MCC Event Register (MCCE)/Mask Register (MCCM)	33-30
32.12.1.1	Interrupt Table Entry	33-31
32.13	MCC Buffer Descriptors	33-32
32.13.1	Receive Buffer Descriptor (RxBD)	33-33
32.13.2	Transmit Buffer Descriptor (TxBD)	33-35
32.14	MCC Initialization and Start/Stop Sequence	33-37
32.14.1	Single-Channel Initialization	33-37
32.14.2	Super Channel Initialization	33-38
32.15	MCC Latency and Performance	33-38

Chapter 33
Fast Communications Controllers (FCCs)

33.1	Overview	34-2
33.2	General FCC Mode Registers (GFMRx)	34-3
33.3	General FCC Expansion Mode Register (GFEMR)	34-7
33.4	FCC Protocol-Specific Mode Registers (FPSMRx)	34-8
33.5	FCC Data Synchronization Registers (FDSRx)	34-8
33.6	FCC Transmit-on-Demand Registers (FTODRx)	34-9
33.7	FCC Buffer Descriptors	34-9
33.8	FCC Parameter RAM	34-11
33.8.1	FCC Function Code Registers (FCRx)	34-13
33.9	Interrupts from the FCCs	34-14
33.9.1	FCC Event Registers (FCCEx)	34-15
33.9.2	FCC Mask Registers (FCCMx)	34-15
33.9.3	FCC Status Registers (FCCSx)	34-15
33.10	FCC Initialization	34-15
33.11	FCC Interrupt Handling	34-16
33.12	FCC Timing Control	34-17

Contents

Paragraph Number	Title	Page Number
33.13	Disabling the FCCs On the Fly	34-20
33.13.1	FCC Transmitter Full Sequence	34-21
33.13.2	FCC Transmitter Shortcut Sequence	34-21
33.13.3	FCC Receiver Full Sequence	34-21
33.13.4	FCC Receiver Shortcut Sequence	34-22
33.13.5	Switching Protocols	34-22
33.14	Saving Power	34-22

Chapter 34
ATM Controller

34.1	Features	35-2
34.2	ATM Controller Overview	35-5
34.2.1	Transmitter Overview	35-5
34.2.1.1	AAL5 Transmitter Overview	35-6
34.2.1.2	AAL1 Transmitter Overview	35-6
34.2.1.3	AAL0 Transmitter Overview	35-6
34.2.1.4	Transmit External Rate and Internal Rate Modes	35-6
34.2.2	Receiver Overview	35-7
34.2.2.1	AAL5 Receiver Overview	35-8
34.2.2.2	AAL1 Receiver Overview	35-8
34.2.2.3	AAL0 Receiver Overview	35-9
34.2.3	Performance Monitoring	35-9
34.2.4	ABR Flow Control	35-9
34.3	ATM Pace Control (APC) Unit	35-9
34.3.1	APC Modes and ATM Service Types	35-9
34.3.2	APC Unit Scheduling Mechanism	35-10
34.3.3	Determining the Scheduling Table Size	35-11
34.3.3.1	Determining the Cells Per Slot (CPS) in a Scheduling Table	35-11
34.3.3.2	Determining the Number of Slots in a Scheduling Table	35-12
34.3.4	Determining the Time-Slot Scheduling Rate of a Channel	35-12
34.3.5	ATM Traffic Type	35-12
34.3.5.1	Peak Cell Rate Traffic Type	35-13
34.3.5.2	Determining the PCR Traffic Type Parameters	35-13
34.3.5.3	Peak and Sustain Traffic Type (VBR)	35-13
34.3.5.3.1	Example for Using VBR Traffic Parameters	35-14
34.3.5.3.2	Handling the Cell Loss Priority (CLP)—VBR Type 1 and 2	35-14
34.3.5.4	Peak and Minimum Cell Rate Traffic Type (UBR+)	35-14
34.3.6	Determining the Priority of an ATM Channel	35-15
34.4	VCI/VPI Address Lookup Mechanism	35-15
34.4.1	External CAM Lookup	35-15
34.4.2	Address Compression	35-16

Contents

Paragraph Number	Title	Page Number
34.4.2.1	VP-Level Address Compression Table (VPLT)	35-18
34.4.2.2	VC-Level Address Compression Tables (VCLTs)	35-19
34.4.3	Misinserted Cells	35-20
34.4.4	Receive Raw Cell Queue	35-20
34.5	Available Bit Rate (ABR) Flow Control	35-21
34.5.1	The ABR Model	35-22
34.5.1.1	ABR Flow Control Source End-System Behavior	35-22
34.5.1.2	ABR Flow Control Destination End-System Behavior	35-23
34.5.1.3	ABR Flowcharts	35-23
34.5.2	RM Cell Structure	35-28
34.5.2.1	RM Cell Rate Representation	35-28
34.5.3	ABR Flow Control Setup	35-29
34.6	OAM Support	35-29
34.6.1	ATM-Layer OAM Definitions	35-29
34.6.2	Virtual Path (F4) Flow Mechanism	35-30
34.6.3	Virtual Channel (F5) Flow Mechanism	35-30
34.6.4	Receiving OAM F4 or F5 Cells	35-31
34.6.5	Transmitting OAM F4 or F5 Cells	35-31
34.6.6	Performance Monitoring	35-31
34.6.6.1	Running a Performance Block Test	35-33
34.6.6.2	PM Block Monitoring	35-33
34.6.6.3	PM Block Generation	35-33
34.6.6.4	BRC Performance Calculations	35-34
34.7	User-Defined Cells (UDC)	35-35
34.7.1	UDC Extended Address Mode (UEAD)	35-35
34.8	ATM Layer Statistics	35-36
34.9	ATM-to-TDM Interworking	35-36
34.9.1	Automatic Data Forwarding	35-36
34.9.2	Using Interrupts in Automatic Data Forwarding	35-37
34.9.3	Timing Issues	35-38
34.9.4	Clock Synchronization (SRTS and Adaptive FIFOs)	35-38
34.9.5	Mapping TDM Time Slots to VCs	35-38
34.9.6	CAS Support	35-38
34.9.7	Trunk Condition	35-39
34.9.8	ATM-to-ATM Data Forwarding	35-39
34.10	ATM Memory Structure	35-39
34.10.1	Parameter RAM	35-40
34.10.1.1	Determining UEAD_OFFSET (UEAD Mode Only)	35-42
34.10.1.2	VCI Filtering (VCIF)	35-42
34.10.1.3	Global Mode Entry (GMODE)	35-43
34.10.2	Connection Tables (RCT, TCT, and TCTE)	35-44

Contents

Paragraph Number	Title	Page Number
34.10.2.1	ATM Channel Code	35-45
34.10.2.2	Receive Connection Table (RCT)	35-46
34.10.2.2.1	AAL5 Protocol-Specific RCT	35-48
34.10.2.2.2	AAL5-ABR Protocol-Specific RCT	35-49
34.10.2.2.3	AAL1 Protocol-Specific RCT	35-50
34.10.2.2.4	AAL0 Protocol-Specific RCT	35-52
34.10.2.3	Transmit Connection Table (TCT)	35-53
34.10.2.3.1	AAL5 Protocol-Specific TCT	35-56
34.10.2.3.2	AAL1 Protocol-Specific TCT	35-57
34.10.2.3.3	AAL0 Protocol-Specific TCT	35-58
34.10.2.3.4	VBR Protocol-Specific TCTE	35-58
34.10.2.3.5	UBR+ Protocol-Specific TCTE	35-59
34.10.2.3.6	ABR Protocol-Specific TCTE	35-60
34.10.3	OAM Performance Monitoring Tables	35-63
34.10.4	APC Data Structure	35-64
34.10.4.1	APC Parameter Tables	35-65
34.10.4.2	APC Priority Table	35-66
34.10.4.3	APC Scheduling Tables	35-66
34.10.5	ATM Controller Buffer Descriptors (BDs)	35-67
34.10.5.1	Transmit Buffer Operation	35-67
34.10.5.2	Receive Buffer Operation	35-68
34.10.5.2.1	Static Buffer Allocation	35-68
34.10.5.2.2	Global Buffer Allocation	35-69
34.10.5.2.3	Free Buffer Pools	35-70
34.10.5.2.4	Free Buffer Pool Parameter Tables	35-71
34.10.5.3	ATM Controller Buffers	35-72
34.10.5.4	AAL5 RxBD	35-72
34.10.5.5	AAL1 RxBD	35-74
34.10.5.6	AAL0 RxBD	35-75
34.10.5.7	AAL5, AAL1 User-Defined Cell—RxBD Extension	35-76
34.10.5.8	AAL5 TxBDs	35-77
34.10.5.9	AAL1 TxBDs	35-78
34.10.5.10	AAL0 TxBDs	35-79
34.10.5.11	AAL5, AAL1 User-Defined Cell—TxBD Extension	35-81
34.10.6	AAL1 Sequence Number (SN) Protection Table (AAL1 Only)	35-81
34.10.7	UNI Statistics Table	35-82
34.11	ATM Exceptions	35-82
34.11.1	Interrupt Queues	35-82
34.11.2	Interrupt Queue Entry	35-83
34.11.3	Interrupt Queue Parameter Tables	35-84
34.12	The UTOPIA Interface	35-85

Contents

Paragraph Number	Title	Page Number
34.12.1	UTOPIA Interface Master Mode	35-85
34.12.1.1	UTOPIA Master Multiple PHY Operation	35-86
34.12.2	UTOPIA Interface Slave Mode	35-86
34.12.2.1	UTOPIA Slave Multiple PHY Operation	35-88
34.12.2.2	UTOPIA Clocking Modes	35-88
34.12.2.3	UTOPIA Loop-Back Modes	35-88
34.13	ATM Registers	35-88
34.13.1	General FCC Mode Register (GFMR)	35-88
34.13.2	FCC Protocol-Specific Mode Register (FPSMR)	35-88
34.13.3	ATM Event Register (FCCE)/ Mask Register (FCCM)	35-90
34.13.4	FCC Transmit Internal Rate Mode	35-92
34.13.5	FCC Transmit Internal Rate Port Enable Register (FIRPER)	35-92
34.13.6	FCC Internal Rate Event Register (FIRER)	35-93
34.13.7	FCC Internal Rate Selection Registers (FIRSR_HI, FIRSR_LO)	35-94
34.13.8	FCC Transmit Internal Rate Register (FTIRRx)	35-95
34.13.8.1	Example	35-96
34.13.9	Internal Rate Programming Model	35-97
34.14	ATM Transmit Command	35-97
34.15	SRTS Generation and Clock Recovery Using External Logic	35-98
34.16	Configuring the ATM Controller for Maximum CPM Performance	35-100
34.16.1	Using Transmit Internal Rate Mode	35-100
34.16.2	APC Configuration	35-101
34.16.3	Buffer Configuration	35-101
34.16.4	Compression and Connection Tables	35-102
34.16.5	Level 2 Cache Utilization	35-102

Chapter 35
AAL1 Circuit Emulation Service

35.1	Features	36-1
35.2	AAL1 Transmitter Overview	36-3
35.2.1	Data Path	36-3
35.2.2	Signaling Path	36-4
35.3	AAL1 Receiver Overview	36-4
35.4	Interworking Functions	36-6
35.4.1	Automatic Data Forwarding	36-7
35.4.1.1	ATM-to-TDM	36-7
35.4.1.2	TDM-to-ATM	36-8
35.4.2	Timing Issues	36-9
35.4.3	Clock Synchronization (SRTS, Adaptive FIFO)	36-10
35.4.4	Mapping TDM Time Slots to VCs	36-10

Contents

Paragraph Number	Title	Page Number
35.4.5	Trunk Condition	36-11
35.4.6	Channel Associated Signaling (CAS) Support	36-11
35.4.7	Mapping VC Signaling to CAS Blocks	36-12
35.4.7.1	CAS Routing Table	36-13
35.4.7.2	TDM-to-ATM CAS Support	36-14
35.4.7.2.1	CAS Mapping Using the Core (Optional)	36-15
35.4.7.3	ATM-to-TDM CAS Support	36-15
35.4.7.3.1	CAS Updates Using the Core (Optional)	36-16
35.5	ATM-to-TDM Adaptive Slip Control	36-16
35.5.1	Pre-Underrun	36-17
35.5.2	Pre-Overrun	36-18
35.5.3	CES Adaptive Threshold Tables	36-21
35.6	3-Step-SN Algorithm	36-22
35.6.1	The Three States of the Algorithm	36-23
35.7	Pointer Verification Mechanism	36-24
35.8	AAL-1 Memory Structure	36-25
35.8.1	AAL1 Parameter RAM	36-25
35.9	Receive and Transmit Connection Tables (RCT, TCT)36-29	
35.9.1	Receive Connection Table (RCT)	36-29
35.9.1.1	AAL1 Protocol-Specific RCT	36-32
35.9.2	Transmit Connection Table (TCT)	36-34
35.9.2.1	AAL1 Protocol-Specific TCT	36-38
35.10	Outgoing CAS Status Register (OCASSR)	36-40
35.11	Buffer Descriptors	36-40
35.11.1	Transmit Buffer Operation	36-41
35.11.2	Receive Buffer Operation	36-41
35.12	ATM Controller Buffers	36-42
35.12.1	AAL1 RxBD	36-43
35.12.2	AAL1 TxBDs	36-44
35.13	AAL1 Exceptions	36-46
35.13.1	AAL1 Interrupt Queue Entry	36-46
35.14	AAL1 Sequence Number (SN) Protection Table	36-47
35.15	Internal AAL1 Statistics Tables	36-48
35.16	External AAL1 Statistics Tables	36-48
35.17	SRTS Generation and Clock Recovery Using External SRTS Logic	36-49
35.18	CES-Specific Additions to the MCC	36-51
35.18.1	CES Additions to the MCC Parameter RAM	36-51
35.18.2	Channel Mode Register (CHAMR)—CES Mode	36-51
35.18.3	Interrupt Table and Mask (INTMSK)	36-53
35.19	Application Considerations	36-53

Contents

Paragraph Number	Title	Page Number
	Chapter 36 AAL2	
36.1	Introduction	37-1
36.2	Features	37-3
36.3	AAL2 Transmitter	37-5
36.3.1	Transmitter Overview	37-5
36.3.2	Transmit Priority Mechanism	37-6
36.3.2.1	Round Robin Priority	37-6
36.3.2.2	Fixed Priority	37-7
36.3.3	Partial Fill Mode (PFM)	37-7
36.3.4	No STF Mode	37-9
36.3.5	AAL2 Tx Data Structures	37-10
36.3.5.1	AAL2 Transmit Connection Tables	37-10
36.3.5.2	CPS Tx Queue Descriptor	37-13
36.3.5.3	CPS Buffer Structure	37-15
36.3.5.4	SSSAR Tx Queue Descriptor	37-17
36.3.5.5	SSSAR Transmit Buffer Descriptor	37-20
36.4	AAL2 Receiver	37-21
36.4.1	Receiver Overview	37-21
36.4.2	Mapping of PHY | VP | VC | CID	37-23
36.4.3	AAL2 Switching	37-24
36.4.4	AAL2 RX Data Structures	37-25
36.4.4.1	AAL2 Receive Connection Tables	37-25
36.4.4.2	CID Mapping Tables and their RxQDs	37-28
36.4.4.3	CPS Rx Queue Descriptors	37-29
36.4.4.4	CPS Switch Rx Queue Descriptor	37-29
36.4.4.5	CPS Receive Buffer Descriptor (RxBD)	37-30
36.4.4.6	SSSAR Rx Queue Descriptor	37-31
36.4.4.7	SSSAR Receive Buffer Descriptor	37-33
36.5	AAL2 Parameter RAM	37-35
36.6	User Defined Cells in AAL2	37-37
36.7	AAL2 Exceptions	37-38
	Chapter 37 Transmission Convergence Layer	
37.1	Features	38-1
37.2	TC Layer Block Diagram	38-3
37.3	Signals	38-4
37.4	Receive ATM Cell Functions	38-4
37.5	Receive ATM 2-Cell FIFO	38-6

Contents

Paragraph Number	Title	Page Number
37.6	Transmit ATM Cell Functions	38-6
37.7	Transmit ATM 2-Cell FIFO	38-7
37.8	Rx UTOPIA Interface	38-7
37.9	Tx UTOPIA Interface	38-7
37.10	TC Layer Connection to FCC2	38-7
37.11	TC Layer Programming Model	38-8
37.11.1	TC Layer Registers	38-8
37.11.1.1	TC Layer Mode Register (TCMODE)	38-8
37.11.1.2	Cell Delineation State Machine Register (CDSMRx)	38-10
37.11.1.3	TC Layer Event Register (TCERx)	38-10
37.11.1.4	TC Layer Mask Register (TCMRx)	38-11
37.11.1.5	CPM Low Interrupt Priority Register (SCPRR_L)	38-11
37.11.2	TC Layer General Registers	38-12
37.11.2.1	TC Layer General Event Register (TCGER)	38-12
37.11.2.2	TC Layer General Status Register (TCGSR)	38-13
37.11.3	TC Layer Cell Counters	38-14
37.11.3.1	Received Cell Counter (RCC)	38-14
37.11.3.2	Transmitted Cell Counter (TCC)	38-14
37.11.3.3	Errorred Cell Counter (ECC)	38-14
37.11.3.4	Corrected Cell Counter (CCC)	38-14
37.11.3.5	Tx IDLE Cell Counter (ICC)	38-14
37.11.3.6	Filtered Cell Counter (FCC)	38-14
37.11.4	Programming FCC2	38-14
37.11.5	Programming and Operating the TC Layer	38-15
37.12	TC Layer Implementation Example	38-17
37.13	Operating the TC Layer at Higher Frequencies	38-18
37.14	Programming a T1 Application	38-18
37.14.1	Step 1	38-19
37.14.2	Step 2	38-19
37.14.3	Step 3	38-19
37.14.4	Step 4	38-19
37.14.5	Step 5	38-20
37.14.6	Step 6	38-20
37.14.7	Step 7	38-20

Chapter 38
CPM Ethernet Controller

38.1	Ethernet Protocol Basics	39-2
38.2	Fast Ethernet on the MPC8560 CPM	39-3
38.3	Features	39-3
38.4	Connecting the MPC8560 to Fast Ethernet	39-5

Contents

Paragraph Number	Title	Page Number
38.4.1	Connecting the MPC8560 to Ethernet (RMII)	39-6
38.5	Ethernet Channel Frame Transmission	39-6
38.6	Ethernet Channel Frame Reception	39-7
38.7	Flow Control	39-8
38.8	CAM Interface	39-9
38.9	Ethernet Parameter RAM	39-10
38.10	Programming Model	39-13
38.11	Ethernet Command Set	39-13
38.12	RMON Support	39-15
38.13	Ethernet Address Recognition	39-16
38.14	Hash Table Algorithm	39-18
38.15	Interpacket Gap Time	39-19
38.16	Handling Collisions	39-19
38.17	Internal and External Loopback	39-19
38.18	Ethernet Error-Handling Procedure	39-20
38.18.1	FCC Ethernet Mode Register (FPSMR)	39-21
38.18.2	Ethernet Event Register (FCCE)/Mask Register (FCCM)	39-23
38.19	Ethernet RxBDs	39-25
38.20	Ethernet TxBDs	39-28

Chapter 39
FCC HDLC Controller

39.1	Key Features	40-2
39.2	HDLC Channel Frame Transmission Processing	40-2
39.3	HDLC Channel Frame Reception Processing	40-3
39.4	HDLC Parameter RAM	40-4
39.5	Programming Model	40-5
39.5.1	HDLC Command Set	40-5
39.5.2	HDLC Error Handling	40-6
39.6	HDLC Mode Register (FPSMR)	40-8
39.7	HDLC Receive Buffer Descriptor (RxBD)	40-9
39.8	HDLC Transmit Buffer Descriptor (TxBD)	40-12
39.9	HDLC Event Register (FCCE)/ Mask Register (FCCM)	40-14
39.10	FCC Status Register (FCCS)	40-16

Chapter 40
FCC Transparent Controller

40.1	Features	41-1
40.2	Transparent Channel Operation	41-2
40.3	Achieving Synchronization in Transparent Mode	41-2

Contents

Paragraph Number	Title	Page Number
40.3.1	In-Line Synchronization Pattern	41-3
40.3.2	External Synchronization Signals	41-3
40.3.3	Transparent Synchronization Example	41-4

Chapter 41
Serial Peripheral Interface (SPI)

41.1	Features	42-1
41.2	SPI Clocking and Signal Functions	42-2
41.3	Configuring the SPI Controller	42-3
41.3.1	The SPI as a Master Device	42-3
41.3.2	The SPI as a Slave Device	42-4
41.3.3	The SPI in Multimaster Operation	42-4
41.4	Programming the SPI Registers	42-7
41.4.1	SPI Mode Register (SPMODE)	42-7
41.4.1.1	SPI Examples with Different SPMODE[LEN] Values	42-9
41.4.2	SPI Event/Mask Registers (SPIE/SPIM)	42-10
41.4.3	SPI Command Register (SPCOM)	42-11
41.5	SPI Parameter RAM	42-11
41.5.1	Receive/Transmit Function Code Registers (RFCR/TFCR)	42-13
41.6	SPI Commands	42-13
41.7	The SPI Buffer Descriptor (BD) Table	42-14
41.7.1	SPI Buffer Descriptors (BDs)	42-14
41.7.1.1	SPI Receive BD (RxBD)	42-15
41.7.1.2	SPI Transmit BD (TxBD)	42-16
41.8	SPI Master Programming Example	42-17
41.9	SPI Slave Programming Example	42-18
41.10	Handling Interrupts in the SPI	42-19

Chapter 42
CPM I^2C Controller

42.1	Features	43-2
42.2	I^2C Controller Clocking and Signal Functions	43-2
42.3	I^2C Controller Transfers	43-3
42.3.1	I^2C Master Write (Slave Read)	43-4
42.3.2	I^2C Loopback Testing	43-4
42.3.3	I^2C Master Read (Slave Write)	43-4
42.3.4	I^2C Multi-Master Considerations	43-5
42.4	I^2C Registers	43-6
42.4.1	I^2C Mode Register (I2MOD)	43-6
42.4.2	I^2C Address Register (I2ADD)	43-7

Contents

Paragraph Number	Title	Page Number
42.4.3	I^2C Baud Rate Generator Register (I2BRG)	43-7
42.4.4	I^2C Event/Mask Registers (I2CER/I2CMR)	43-8
42.4.5	I^2C Command Register (I2COM)	43-8
42.5	I^2C Parameter RAM	43-9
42.6	I^2C Commands	43-11
42.7	I^2C Buffer Descriptor (BD) Tables	43-12
42.8	I^2C Buffer Descriptors (BDs)	43-12
42.8.0.1	I^2C Receive Buffer Descriptor (RxBD)	43-13
42.8.0.2	I^2C Transmit Buffer Descriptor (TxBD)	43-14

Chapter 43
Parallel I/O Ports

43.1	Features	44-1
43.2	Port Registers	44-2
43.2.1	Port Open-Drain Registers (PODRA–PODRD)	44-2
43.2.2	Port Data Registers (PDATA–PDATD)	44-2
43.2.3	Port Data Direction Registers (PDIRA–PDIRD)	44-3
43.2.4	Port Pin Assignment Register (PPAR)	44-4
43.2.5	Port Special Options Registers A–D (PSORA–PSORD)	44-4
43.3	Port Block Diagram	44-5
43.4	Port Pins Functions	44-6
43.4.1	General Purpose I/O Pins	44-6
43.4.2	Dedicated Pins	44-7
43.5	Ports Tables	44-7
43.6	Interrupts from Port C	44-20

Chapter 44
CPM ROM Microcode: SS7 Controller

44.1	SS7 Data Structure Organization	45-1
44.2	Global SS7 Parameters	45-1
44.3	Channel Extra Parameters	45-1
44.4	Super Channel Table	45-1
44.5	TDM Mapping	45-1
44.6	SS7 Commands	45-1
44.7	SS7 Exceptions	45-2
44.7.1	Interrupt Table Entry	45-2
44.8	SS7 Buffer Descriptors	45-3
44.8.1	SS7 Receive Buffer Descriptor (RxBD)	45-3
44.8.2	SS7 Transmit Buffer Descriptor (TxBD)	45-6
44.9	SS7 Initialization and Start/Stop Sequence	45-7

Part I
Overview

Part I describes the many features of the MPC8560 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 MPC8560 integrated communications controller. It describes the MPC8560, its interfaces, and its programming model. This chapter describes the functional operation of the MPC8560 with emphasis on peripheral functions.

Chapter 2, “Memory Map,” describes the memory map of the MPC8560. 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 MPC8560.

Chapter 1
Overview

The MPC8560 PowerQUICC III is a next-generation PowerQUICC II integrated communications processor. The MPC8560 provides integration of processing power for networking and communications peripherals resulting in higher device performance.The MPC8560 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 MPC8560 integrated microprocessor.

1.1 Introduction

Motorola's leading PowerQUICC III architecture integrates two processing blocks. One block of the MPC8560 is a high-performance embedded e500 core. 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.

A second block is the communications processor module (CPM). The CPM of the MPC8560 supports three fast serial communications channels (FCCs) for 155 Mbps ATM and Fast Ethernet, and up to 256 full-duplex, time-division-multiplexed (TDM) channels using two multi-channel controllers (MCCs). In addition, the CPM of the PowerQUICC III supports four serial communications controllers (SCCs), one serial peripheral interface (SPI), and one I^2C interface.

In addition, the MPC8560 offers two integrated 10/100 Mbps/1 Gbps three speed Ethernet controllers (TSECs), a DDR SDRAM memory controller, a 64-bit PCI/PCI-X controller, and an 8-bit RapidIO port, a programmable interrupt controller, an I^2C controller, a 4-channel DMA controller, and a general purpose I/O port. The high level of integration in the MPC8560 helps simplify board design and offers significant bandwidth and performance for high-end control-plane and data-plane applications.

1.2 MPC8560 Overview

The following section provides a high-level overview of the MPC8560 features.

Figure 1-1 shows the major functional units within the MPC8560.

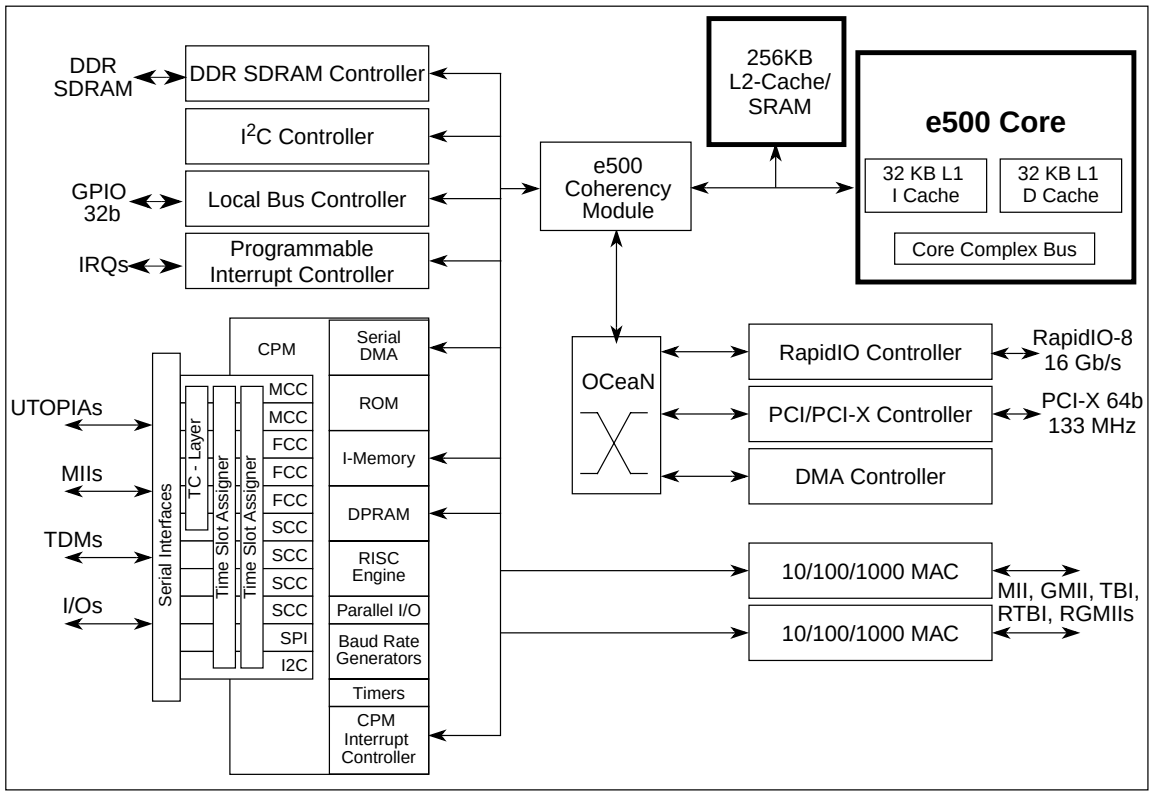


Figure 1-1. MPC8560 Block Diagram

1.2.1 Key Features

The following lists an overview of the MPC8560 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
- High-performance RISC CPM operating at up to 333 MHz
  - — CPM software compatibility with previous PowerQUICC families
  - — One instruction per clock
  - — Executes code from internal ROM or instruction RAM
  - — 32-bit RISC architecture
  - — Tuned for communication environments: instruction set supports CRC computation and bit manipulation.
  - — Internal timer
  - — Interfaces with the embedded e500 core processor through a 32-Kbyte dual-port RAM and virtual DMA channels for each peripheral controller
  - — Handles serial protocols and virtual DMA.
  - — Three full-duplex fast serial communications controllers (FCCs) support the following protocols:
    - – ATM protocol through UTOPIA interface (FCC1 and FCC2 only)
    - – IEEE802.3/Fast Ethernet
    - – HDLC
    - – Totally transparent operation
  - — Two multi-channel controllers (MCCs) that together can handle up to 256 HDLC/transparent channels at 64 Kbps each, multiplexed on up to eight TDM interfaces
  - — Four full-duplex serial communications controllers (SCCs) support the following protocols:
    - – High level/synchronous data link control (HDLC/SDLC)
    - – LocalTalk (HDLC-based local area network protocol)
    - – Universal asynchronous receiver transmitter (UART)
    - – Synchronous UART (1x clock mode)
    - – Binary synchronous communication (BISYNC)
    - – Totally transparent operation
  - — Serial peripheral interface (SPI) support for master or slave
  - — I^2C bus controller
  - — Time-slot assigner supports multiplexing of data from any of the SCCs and FCCs onto eight time-division multiplexed (TDM) interfaces. The time-slot assigner supports the following TDM formats:

  - T1/CEPT lines
  - T3/E3
  - Pulse code modulation (PCM) highway interface
  - ISDN primary rate
  - Motorola interchip digital link (IDL)
  - General circuit interface (GCI)
- User-defined interfaces
- Eight independent baud rate generators (BRGs)
- Four general-purpose 16-bit timers or two 32-bit timers
- General-purpose parallel ports—16 parallel I/O lines with interrupt capability
- Supports inverse muxing of ATM cells (IMA)

- 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 discontiguous 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
- I^2C controller
  - Two-wire interface
  - Multiple master support
  - Master or slave I^2C mode support
  - On-chip digital filtering rejects spikes on the bus
- Boot sequencer
  - Optionally loads configuration data from serial ROM at reset via the I^2C interface
  - Can be used to initialize configuration registers and/or memory
  - Supports extended I^2C addressing mode
  - Data integrity checked with preamble signature and CRC
- 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, 820.3x, 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 MPC8560 Architecture Overview

The following section describe the major functional units of the MPC8560.

1.3.1 e500 Core Overview

The MPC8560 uses the e500 microprocessor core complex. Both the e500 core and the CPM have an internal PLL, that allows independent optimization of their operating frequencies. The core and CPM 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 Communications Processor Module (CPM)

The CPM contains features that allow the MPC8560 to excel in a variety of applications targeted for networking and telecommunication markets. The MPC8560 CPM is a superset of the MPC8260 PowerQUICC II, with enhanced CP performance. The CPM also has additional hardware and microcode routines that support high bit rate protocols like ATM (up to 155 Mbps full-duplex) and Fast Ethernet (100 Mbps full-duplex).

Figure 1-2 shows the major functional units within the MPC8560 CPM.

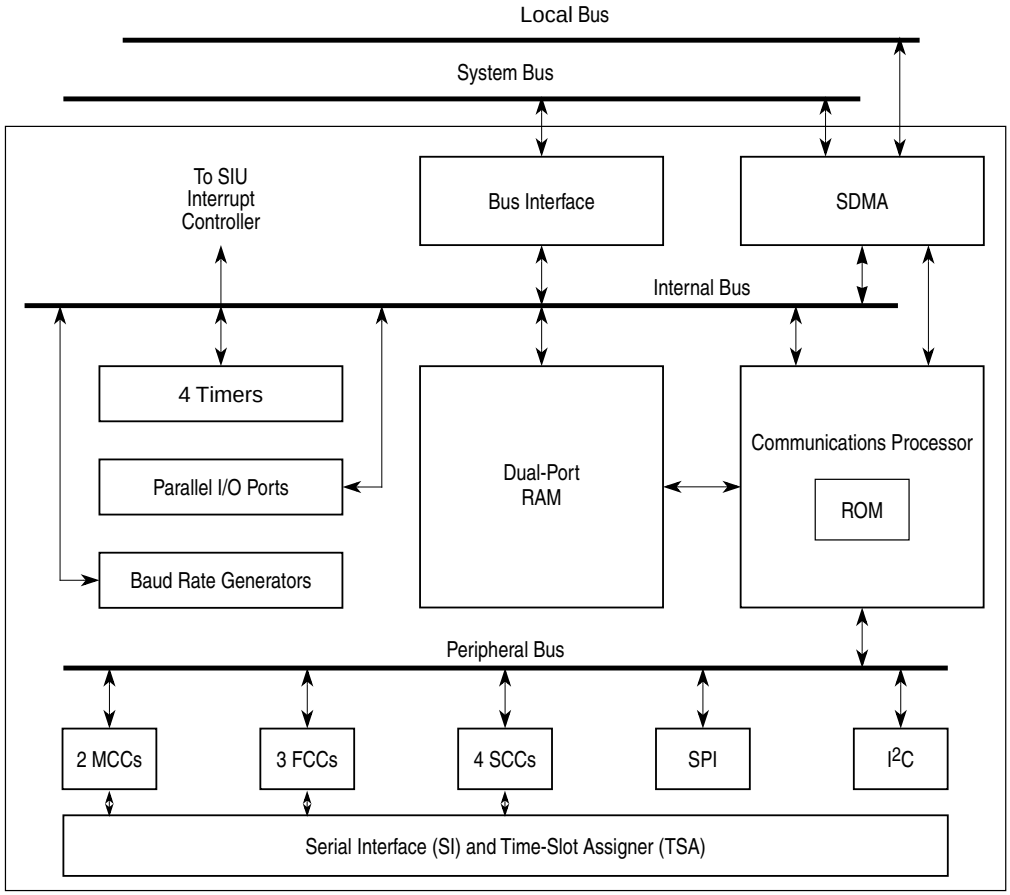


Figure 1-2. MPC8560 CPM Block Diagram

The following list summarizes the major features of the CPM:

- The communications processor (CP) is an embedded 32-bit RISC controller residing on a separate bus (CPM local bus). With this separate bus, the CP does not affect the performance of the e500 core. The CP handles the lower-layer tasks and DMA control activities, leaving the e500 core free to handle higher-layer activities. The CP has an instruction set optimized for communications but that can also be used for general-purpose applications, relieving the system core of small, often repeated tasks.
- Two serial DMAs (SDMAs), one associated with the local bus and one associated with the coherency module, handling transfers simultaneously
- Three full-duplex, serial fast communications controllers (FCCs) supporting ATM (155 Mbps) protocol through UTOPIA level 2 interface (there are two UTOPIA interfaces on the MPC8560), IEEE 802.3 and Fast Ethernet protocols, HDLC up to E3 rates (45 Mbps) and totally transparent operation. Each FCC can be configured to transmit fully transparent and receive HDLC data or vice-versa.

- Two multichannel controllers (MCCs) that can handle an aggregate of 256 x 64 Kbps HDLC or transparent channels, multiplexed on up to eight TDM interfaces. The MCC also supports super-channels of rates higher than 64 Kbps and subchanneling of the 64 Kbps channels.
- Four full-duplex serial communications controllers (SCCs) supporting, high-level synchronous data link control, HDLC, local talk, UART, synchronous UART, BISYNC, and transparent.
- Serial peripheral interface (SPI) and I^2C bus controllers
- Time-slot assigner (TSA) that supports multiplexing of data from any of the four SCCs and three FCCs
- ATM TC-layer functionality is implemented internally to support applications which receive ATM traffic over standard serial protocols (T1, E1, xDSL) via its serial interface ports

1.3.3 On-Chip Memory Unit

The MPC8560 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 MPC8560 system logic.

1.3.3.1 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).

1.3.3.2 On-Chip Memory as L2 Cache

The MPC8560 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.

1.3.4 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.

1.3.5 DDR SDRAM Controller

The MPC8560 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 MPC8560 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 MPC8560 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 MPC8560 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 MPC8560 supports error checking and correction (ECC) for system memory. Using ECC, the MPC8560 detects and corrects all single-bit errors and detects all double-bit errors and all errors within a nibble.

The MPC8560 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.

1.3.6 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 MPC8560 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.

1.3.7 I^2C Controller

The inter-IC (IIC or I^2C) bus is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between devices. The synchronous, multi-master bus of the I^2C allows the MPC8560 to exchange data with other I^2C devices such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The two-wire bus (serial data SDA and serial clock SCL) minimizes the interconnections between devices. The synchronous, multimaster bus of the I^2C allows the connection of additional devices to the bus for expansion and system development.

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

1.3.8 Boot Sequencer

The MPC8560 provides an boot sequencer that uses the I^2C interface to access an external serial ROM and loads the data into the MPC8560's configuration registers. The boot sequencer is enabled by a configuration pin sampled when the MPC8560's hardware reset signal is negated. If enabled, the boot sequencer holds The MPC8560'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.

1.3.9 Local Bus Controller

The MPC8560 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, FEPROM, 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 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 transmitt FIFOs, and DMA functions.

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

The MPC8560 TSEC supports programmable CRC generation and checking, RMON statistics, and jumbo frames up to 9.6 Kbytes.

1.3.11 Integrated DMA

The MPC8560 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 $\overline{\text{DMA_DREQ}}$, $\overline{\text{DMA_DACK}}$, and $\overline{\text{DMA_DDONE}}$ 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 MPC8560 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 MPC8560 supports read and write operations to the PCI memory space, the PCI I/O space, and the PCI configuration space. Also, the MPC8560 can generate PCI special-cycle and interrupt-acknowledge commands. As a target, the MPC8560 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 MPC8560 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 MPC8560 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 MPC8560 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 Gpbs.

1.3.13.1 RapidIO Message Unit

The MPC8560’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 MPC8560 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 MPC8560 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 MPC8560 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 MPC8560 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 MPC8560 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 MPC8560 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 Data Processing Overview

Protocol data units (PDUs) can navigate through the various MPC8560 I/O ports in three ways. In the first, data is processed by the MPC8560 CPM (as it is received and transmitted through the UTOPIAs, MIIs, and TDMs associated with the CPM unit) and the local bus. In the second method, data is received by any of the available I/O ports, is sent through the on-chip switch fabric, and is finally transmitted on the target I/O port without the use of the ECM. Lastly, data can be routed from any I/O port to any other I/O port via the ECM.

1.5 Processing Between the CPM and Local Bus

In this case, the MPC8560 stores data in buffers which reside in SDRAM on the local bus. These buffers are each referenced by a buffer descriptor (BD), which may reside in one of two tables (Rx-receive and Tx-transmit) typically placed in DPRAM. The following is a general overview of how incoming data is processed by the CPM (refer to Figure 1-3).

1. When Rx data arrives on the I/O port, it is decoded: the PDU is delineated from the incoming data stream.
2. Next, data is converted from its serial form into a parallel form and then loaded into the Rx FIFO.
3. When the Rx FIFO is filled to a set threshold, the respective communication channel signals the CPM for service.
4. The CPM accesses the next available RxBD in the RxBD table (pointed to by a register in the channel’s parameter RAM table). The BD defines the main memory location where the data is to be placed, as well as the length of this buffer. Data is then moved from the Rx FIFO to a temporary storage location by the CPM.
5. Data is finally moved from this temporary storage location to main memory through DMA transactions. The status and control bits of the BD are updated, and the BD is closed.
6. A CPU interrupt is instantiated to notify the core that a new packet has been received.

Because FIFOs are typically smaller than the incoming PDU, steps 1–3 may be repeated several times to store the entire packet. There may also be times when the incoming PDU

is larger than the buffer length defined in the RxBD. In such cases, steps 4–5 may be repeated and several BDs may be opened and closed to store the entire PDU.

When the transmit portion of the communication channel is enabled, the CPM starts with the first BD in the TxBD table (pointed to by a register in the channel's parameter RAM table), polling the ready R bit of the BD to verify that the buffer is ready for transmission.

7. When the BD is marked ready, the data is moved from the main memory buffer to a temporary storage location through DMA transactions.
8. The CPM moves data from the temporary memory location to the Tx FIFO.
9. Data is taken from the Tx FIFO in its parallel form and serialized.
10. When the Tx FIFO is emptied to a set threshold, the communication channel signals the CPM for more data in order to maintain throughput.
11. The serialized data is lastly encoded and transmitted on the I/O port.

Again, because the buffer pointed to by the Tx BD is typically larger than the Tx FIFO, the communication channel may make several iterations of steps 7–10 to load and transmit the entire PDU.

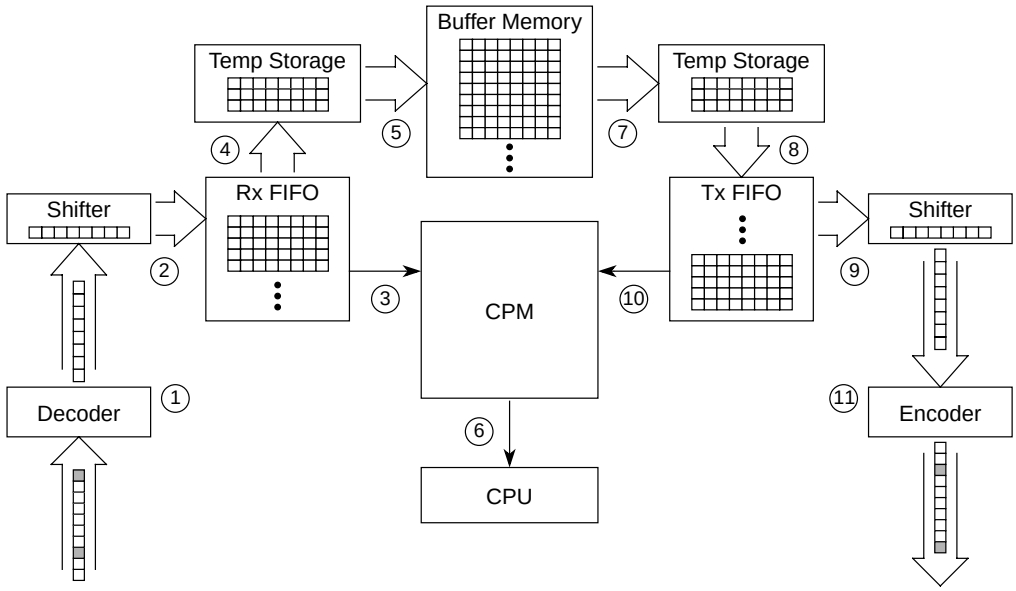


Figure 1-3. Data Processing Within the CPM

1.6 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 MPC8560. 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 1-4).

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.

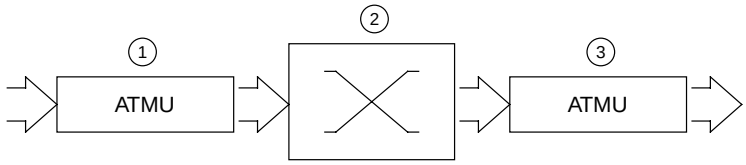


Figure 1-4. Processing Transactions Across the On-Chip Fabric

1.7 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.

While transactions between the CPM and local bus are considered non-coherent because the CPM typically interfaces directly to the local bus (where its buffers are stored), CPM transactions can be made coherent. ATM transactions on a per-connection and direction basis can be set as coherent by setting the necessary bits in the receive and transmit connection tables. Coherency of MCC transactions per logical channel is determined by bits set in TSTATE. FCC and SCC transactions per physical channel and direction can be programmed as coherent by asserting the appropriate bits in the FCC and SCC functional code registers, respectively.

1.8 MPC8560 Application Examples

The following section provides block diagrams of different MPC8560 applications. The MPC8560 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.

1.8.1 Device Configurations

There are three main system configurations for the MPC8560:

- Single-processor system
- Multiprocessor system
- High-performance system

1.8.1.1 Single-Processor System

In this system configuration, shown in Figure 1-5, the MPC8560 core uses the 64-bit DDR SDRAM bus to store data. The 32-bit local bus data is needed to store connection tables for many active ATM connections. The local bus may also be used to store data that does not need to be heavily processed by the e500 core. The CPM) can store large data frames in the local memory without interfering with the operation of the e500 core.

Figure 1-5 shows a basic system configuration.

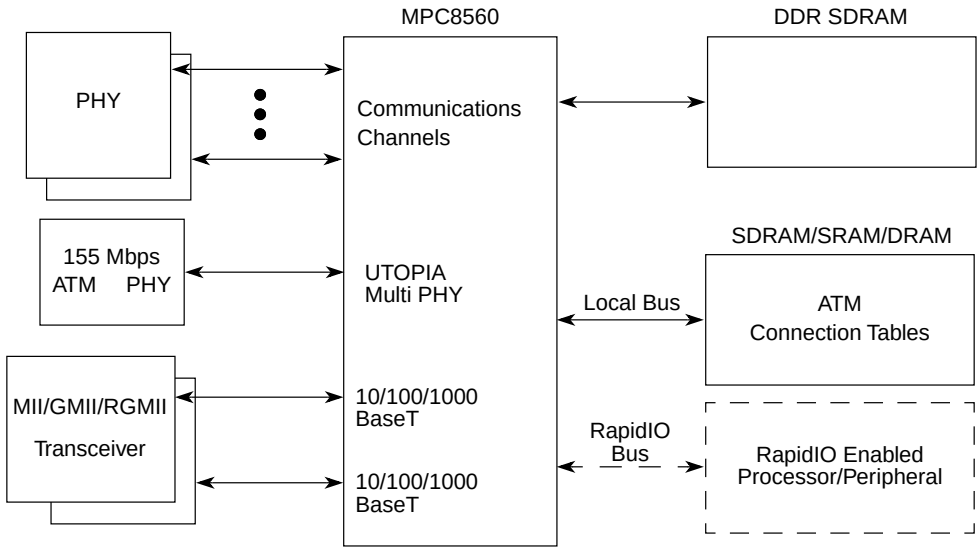


Figure 1-5. Basic System Configuration

1.8.1.2 Multiprocessor System

Figure 1-6 shows a multiprocessor system configuration.

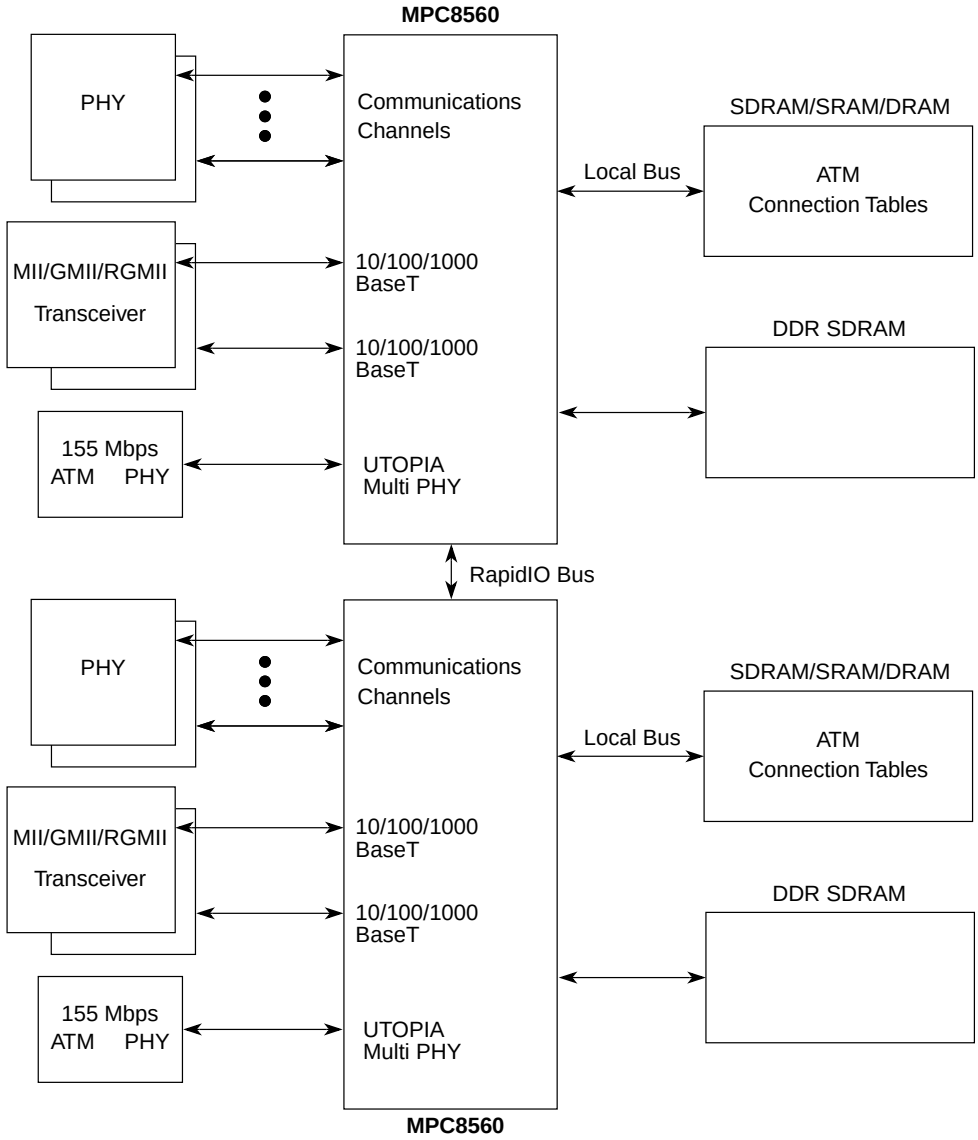


Figure 1-6. High-Performance Communications

This system enhances the serial throughput by connecting one MPC8560 to another MPC8560 with the RapidIO interface. The core in one of the MPC8560 devices can easily access the data stored in the DDR SDRAM memory of the other MPC8560. For performance reasons, the connection table information stored in the local bus memory for one MPC8560 should be copied into the local bus memory of the second. A system of this type can support 512 64-Kbps channels.

1.8.1.3 High-Performance System

Figure 1-7 shows a configuration with a high-performance system.

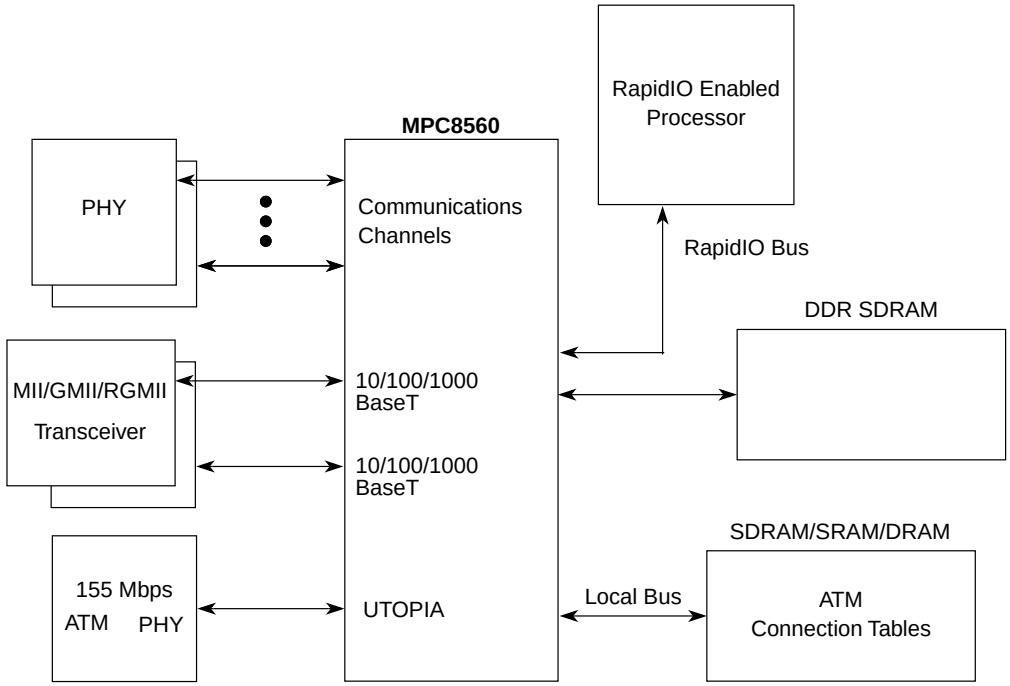


Figure 1-7. High-Performance System Microprocessor Configuration

In this system, an external high-performance microprocessor is connected to the MPC8560 via the RapidIO bus to share the processing load of the e500 core between the MPC8560 and the external processor to increase higher layer processing.

1.8.2 Examples of Communications Systems

Following are some examples of communications systems:

- Remote access server
- Regional office router
- LAN-to-WAN bridge router
- Cellular base station
- 3G wireless base station
- Telecom switch controller
- SONET transmission controller
- Frame relay card
- ATM protocol converter

1.8.2.1 Remote Access Server

See Figure 1-8 for a remote access server configuration.

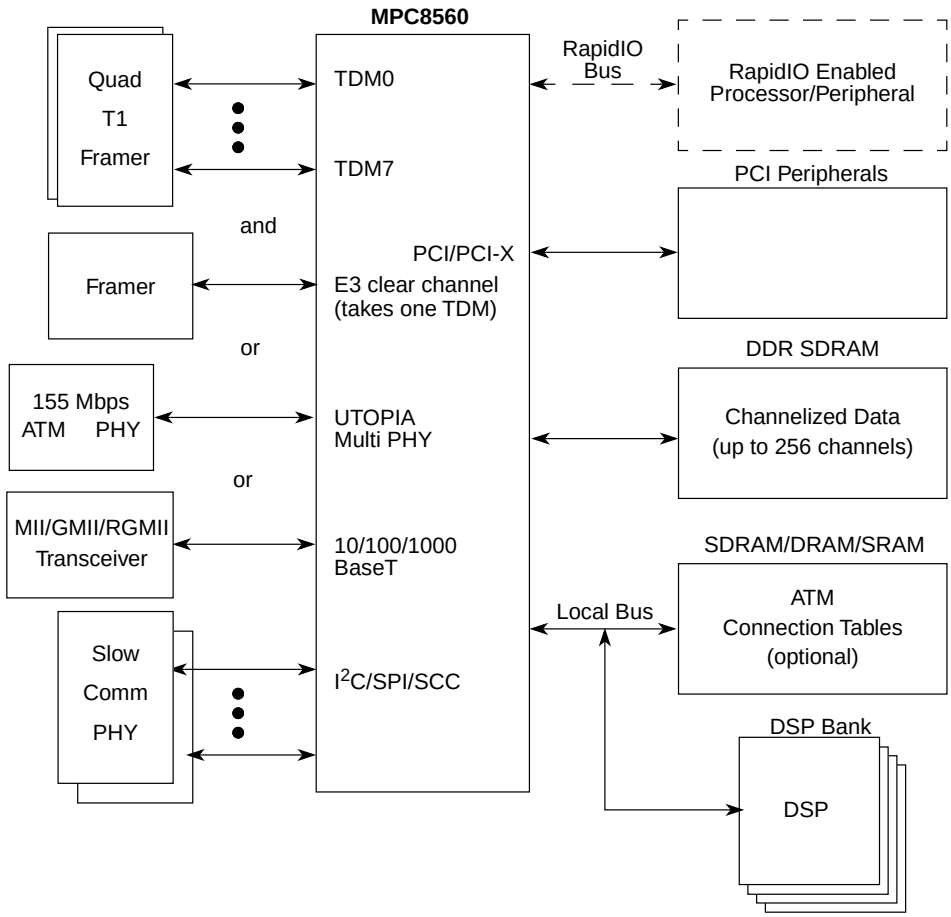


Figure 1-8. Remote Access Server Configuration

In this application, eight TDM ports are connected to external framers. In the MPC8560, each group of four ports support up to 128 channels. One TDM interface can support 32–128 channels. The MPC8560 receives and transmits data in transparent or HDLC mode, and stores or retrieves the channelized data from memory. The data can be stored either in the DDR SDRAM memory or in memory residing on the local bus.

The main trunk can be configured as one of the following:

- 155 Mbps full-duplex ATM, using the UTOPIA interface
- 10/100/1000BaseT Ethernet with MII/GMII/RGMII interface
- High-speed serial channel (up to 45 Mbps)

In ATM mode, there may be a need to store connection tables in external memory on the local bus (if more than 128 active connections are needed). The need for local bus memory depends on the total throughput of the system. The MPC8560 supports automatic (without software intervention) cross connect between ATM and MCC, routing ATM AAL1 frames to MCC slots.

The local bus can be used as an interface to a bank of DSPs that can perform analog modem signal modulation. Data to and from the DSPs can be transferred through the MPC8560 DMA controller.

The MPC8560 local bus memory controller supports pipeline SDRAM devices for efficient burst transfers. A separate DDR controller and bus is available on the MPC8560 to support DDR SDRAM.

1.8.2.2 Regional Office Router

Figure 1-9 shows a regional office router configuration.

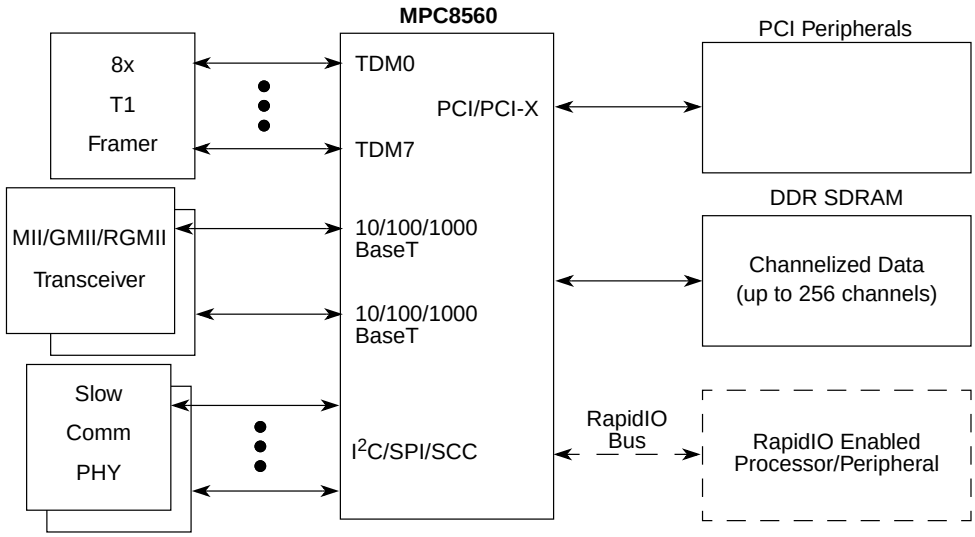


Figure 1-9. Regional Office Router Configuration

In this application, the MPC8560 is connected to up to eight TDM interfaces with up to 256 channels. Each TDM port supports 32–128 channels. If 256 channels are needed, each TDM port can be configured to support 32 channels. In this application there are two MII/GMII/RGMII ports for 10/100/1000BaseT LAN connections.

In all the examples, the SCC ports can be used for management.

1.8.2.3 LAN-to-WAN Bridge Router

Figure 1-10 shows a LAN-to-WAN router configuration, which is similar to the previous example.

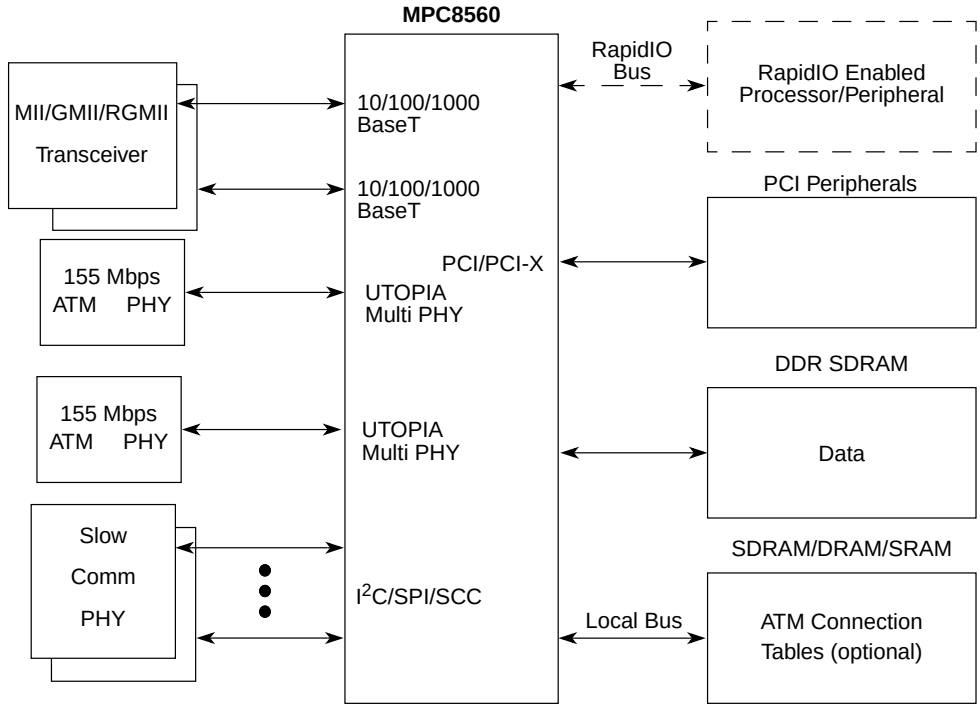


Figure 1-10. LAN-to-WAN Bridge Router Configuration

1.8.2.4 Cellular Base Station

Figure 1-11 shows a cellular base station configuration.

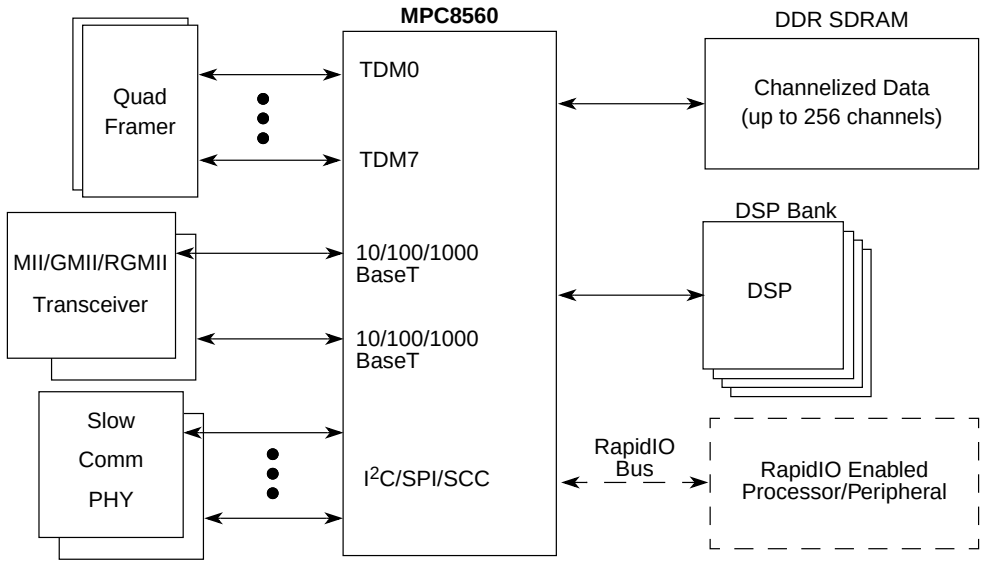


Figure 1-11. Cellular Base Station Configuration

Here the MPC8560 channelizes eight E1s (up to 256, 64-Kbps channels). The local bus can control a bank of DSPs. Data to and from the DSPs can be transferred through the local bus to the host port of the DSPs through the integrated DMA controller. The slower communications ports (SCCs, I^2C, SPI) can be used for management and debug functions.

1.8.2.5 3G Wireless Base Station

Figure 1-12 shows a 3G wireless base station configuration.

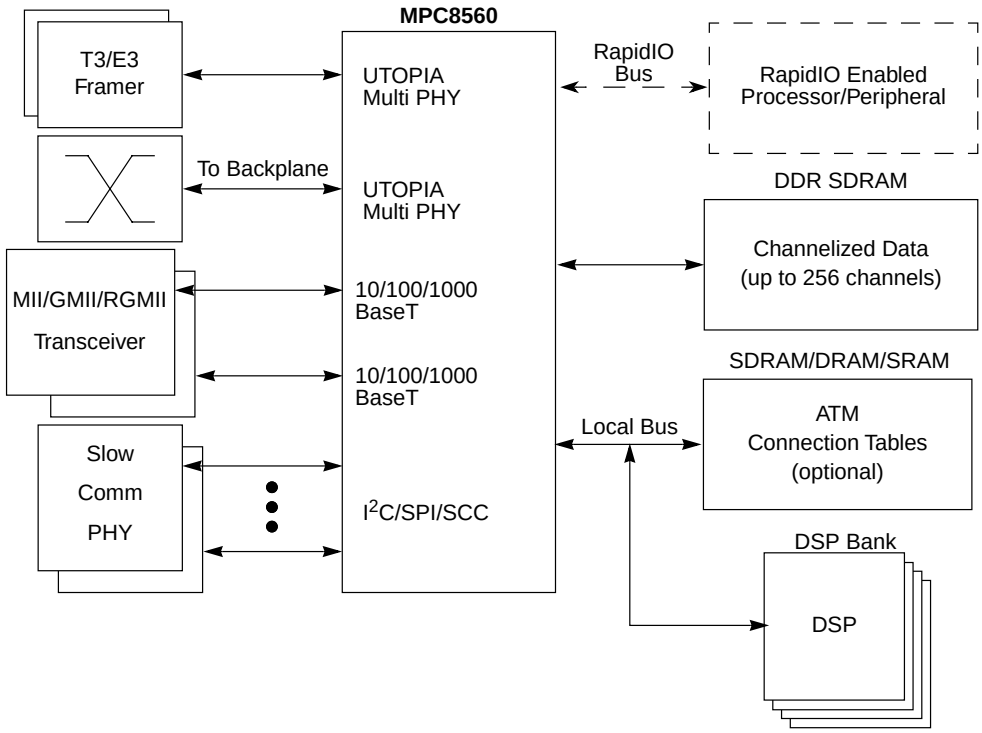


Figure 1-12. 3G Wireless Base Station Configuration

Here the MPC8560 uses two E3/T3s for ATM connections or alternatively two high-bit rate HDLC connections. The local bus can control a bank of DSPs as well as store ATM connection tables. Data to and from the DSPs can be transferred through the local bus to the host port of the DSPs through the integrated DMA controller. The slower communications ports (SCCs, I^2C, SPI) can be used for management and debug functions.

1.8.2.6 Telecommunications Switch Controller

Figure 1-13 shows a telecommunications switch controller configuration.

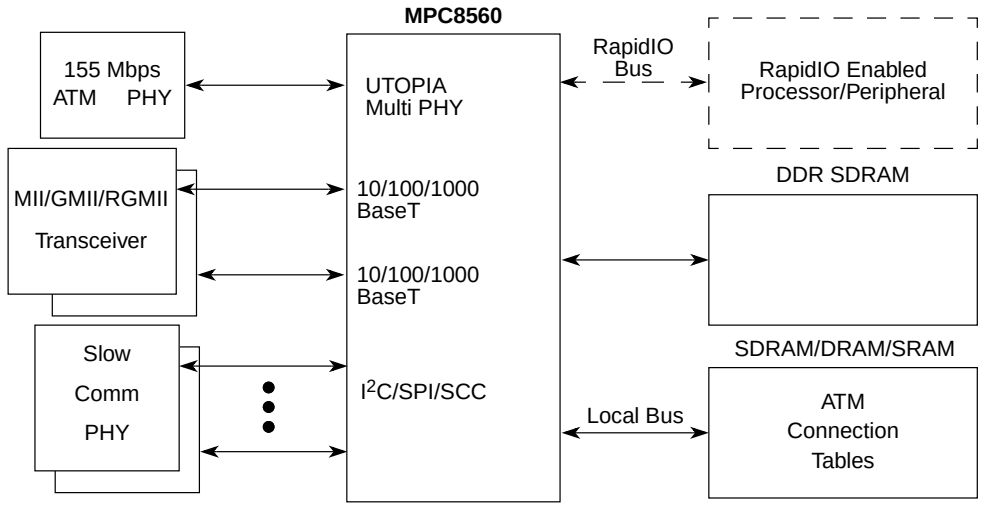


Figure 1-13. Telecommunications Switch Controller Configuration

The MPC8560 CPM supports a total aggregate throughput of 1 Gbps at 333 MHz. This includes one full-duplex, 155-Mbps ATM channel.

1.8.2.7 SONET Transmission Controller

Figure 1-14 shows a SONET transmission controller configuration.

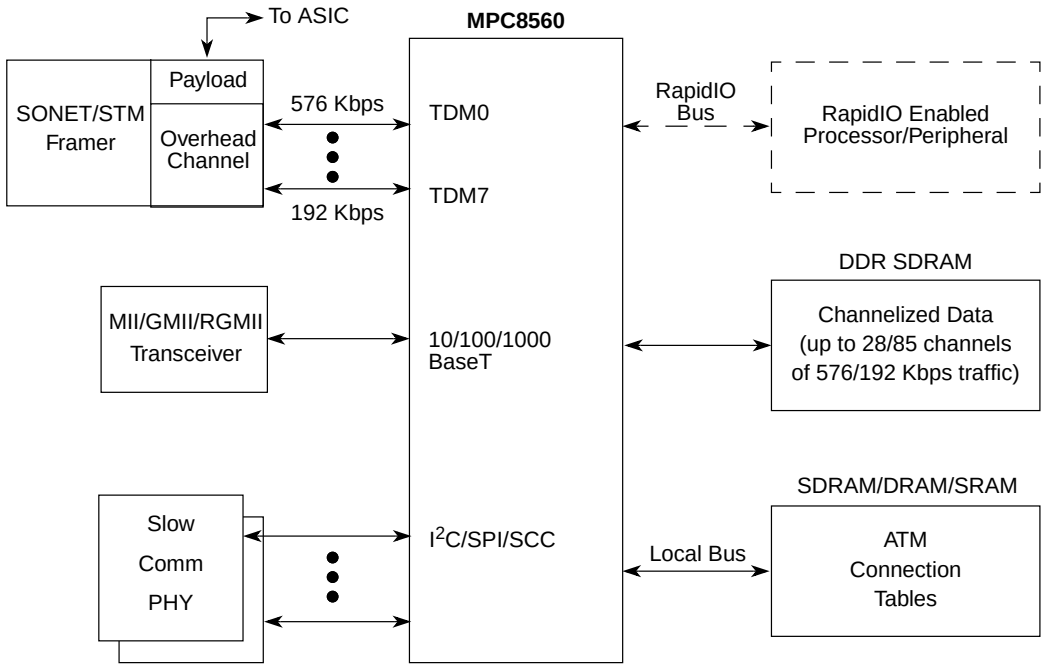


Figure 1-14. SONET Transmission Controller Configuration

In this application, the MPC8560 implements super channeling with the multichannel controller (MCC). Nine 64-Kbps channels are aggregated to form a 576-Kbps channel. The MPC8560 at 333 MHz can support up to 28 576-Kbps superchannels. The MPC8560 also supports subchanneling (under 64 Kbps) with its MCC.

1.8.2.8 Frame Relay Card

Figure 1-15 shows a frame relay card configuration.

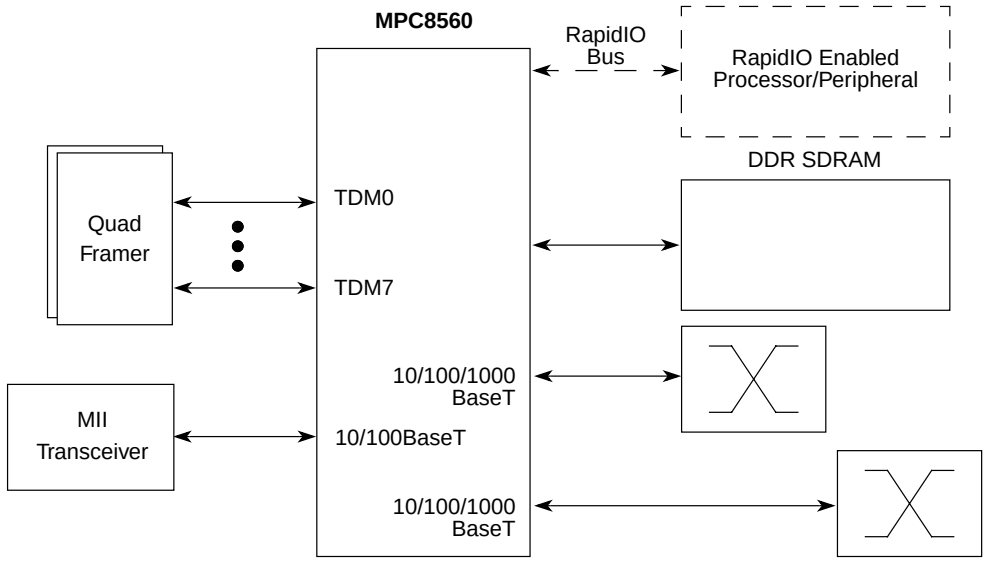


Figure 1-15. Frame Relay Card Configuration

1.8.2.9 ATM Protocol Converter

Figure 1-16 shows an ATM protocol converter.

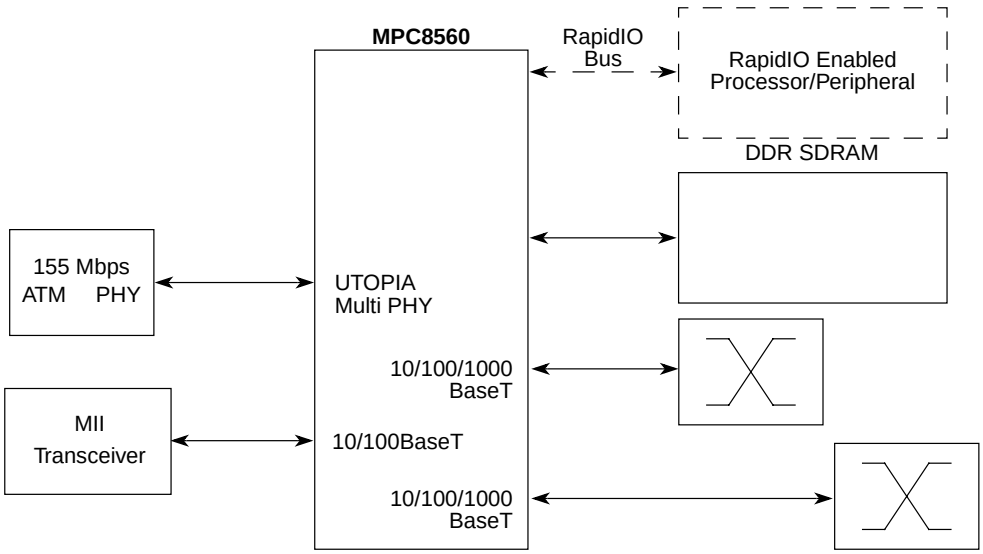


Figure 1-16. ATM Protocol Converter Configuration

In this configuration the MPC8560 can convert traffic from ATM to Ethernet and from Ethernet to ATM. The MPC8560 is also connected to a redundant gigabit Ethernet switch fabric backplane via the two TSECs.

1.9 Compatibility Issues

This section describes some software and hardware compatibility issues.

1.9.1 Software

The MPC8560 CPM features are similar to those in the previous generation MPC8260. The code ports easily from previous devices to the MPC8560, except for new protocols. During the definition of this device, an effort was made to maintain compatibility wherever possible.

Note that the MPC8560 initialization code requires changes from the MPC8260 initialization code (Motorola will provide the reference code.)

1.9.2 MPC8560 Hardware

As the MPC8560 family migrates to smaller geometries, the core voltage will reduce from 1.2 V to low voltages. A programmable voltage regulator is recommended for future

compatibility. See the *MPC8560 Hardware Specifications* for the electrical requirements and the AC and DC characteristics.

1.9.3 Differences Between MPC8560 and MPC8260

While the MPC8560 adds considerably to the functionality offered by the MPC8260, the following MPC8260 features are not included on the MPC8560:

- 60x bus interface
- CPM IDMA support and FlyBy DMA. (An IDMA controller outside the CPM is replacing this.)
- IEEE 802.3/Ethernet support on the SCCs
- SMC functional blocks in the CPM

1.9.4 Communications Protocol Table

Table 1-1 summarizes available protocols for each communications port.

Table 1-1. MPC8560 Protocols

Protocol	Port				
	TSEC	FCC	SCC	MCC	TC Layer
ATM (UTOPIA)		√			
ATM (Serial)					√
1000BaseT	√				
100BaseT	√	√			
10BaseT	√	√			
HDLC		√	√	√	
HDLC_BUS			√		
Transparent		√	√	√	
UART			√		
Multichannel				√	

1.9.5 MPC8560 Configurations

The MPC8560 offers flexibility in configuring the device for specific applications. The functions mentioned in the above sections are all available in the device, but not all of them can be used at the same time. This does not imply that the device is not fully activated in any given implementation. The CPM architecture has the advantage of using common hardware resources for many different protocols, and applications. Two factors limit the functionality in any given system: pinout and performance.

1.9.6 Pin Configurations

To maximize the efficiency of device pins, some pins have multiple functions. In some cases choosing a function may preclude the use of another function.

1.9.7 Communications Performance

The CPM is designed to handle an aggregate of 1 Gbps on the communications channels running at 333 MHz. Performance depends on a number of factors:

- Channel rate versus CPM clock frequency for adequate polling of communications channels for service
- Channel rate and protocol versus CPM clock frequency for CP protocol handling
- Channel rate and protocol versus bus bandwidth
- Channel rate and protocol versus system core clock for adequate protocol handling

The second item above is addressed in this section—the CP's ability to handle high bit-rate protocols. Slow bit-rate protocols do not significantly affect those numbers.

Table 1-2 shows the peak CPM performance of various protocols under the assumption that only one of those protocols is running at a given time. The ATM numbers shown also assume that the local bus is used exclusively by the CPM and enough bandwidth on the DDR memory system is available for the CPM (implying that other resources like the PCI-X controller, TSECs, RapidIO interconnect, DMA controller, and CPU do not all operate at their maximum performance). The frequency specified is the minimum CPM frequency necessary to run the mentioned protocols concurrently in full-duplex.

Table 1-2. MPC8560 Serial Performance

<table>
<tr><th colspan="3" rowspan="2">Protocol</th><th colspan="3">Frame Size</th></tr>
<tr><th>1024 Bytes</th><th>128 Bytes</th><th>64 Bytes</th></tr>
<tr><td>Ethernet</td><td colspan="2">FCC: 100Base T</td><td colspan="3">3 x 100Base T Full Duplex = 600 Mbps</td></tr>
<tr><td>HDLC</td><td colspan="2">MCC: HDLC</td><td colspan="3">256 Full Duplex Channels at 64 Kbps = 16.7 Mbps</td></tr>
<tr><td rowspan="5">ATM</td><td rowspan="2">FCC: AAL5</td><td>No Bus Limitation</td><td colspan="2" rowspan="2">≥ 1000 Mbps Aggregated</td><td rowspan="2">≥ 900 Mbps Aggregated</td></tr>
<tr><td>Connection Tables on Local Bus</td></tr>
<tr><td rowspan="2">FCC: AAL0</td><td>No Bus Limitation</td><td colspan="3" rowspan="2">≥ 1000 Mbps Aggregated</td></tr>
<tr><td>Connection Tables on Local Bus</td></tr>
<tr><td colspan="2">FCC: AAL2 CPS</td><td colspan="3">33–242 Mbps Depending on PDU Size</td></tr>
</table>

These performance estimates assume the CPM is operating at 333 MHz, and that data is stored in SDRAM on the local bus operating at 166 MHz.

Chapter 2
Memory Map

This chapter describes the memory map of the MPC8560. 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 MPC8560 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 MPC8560'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.

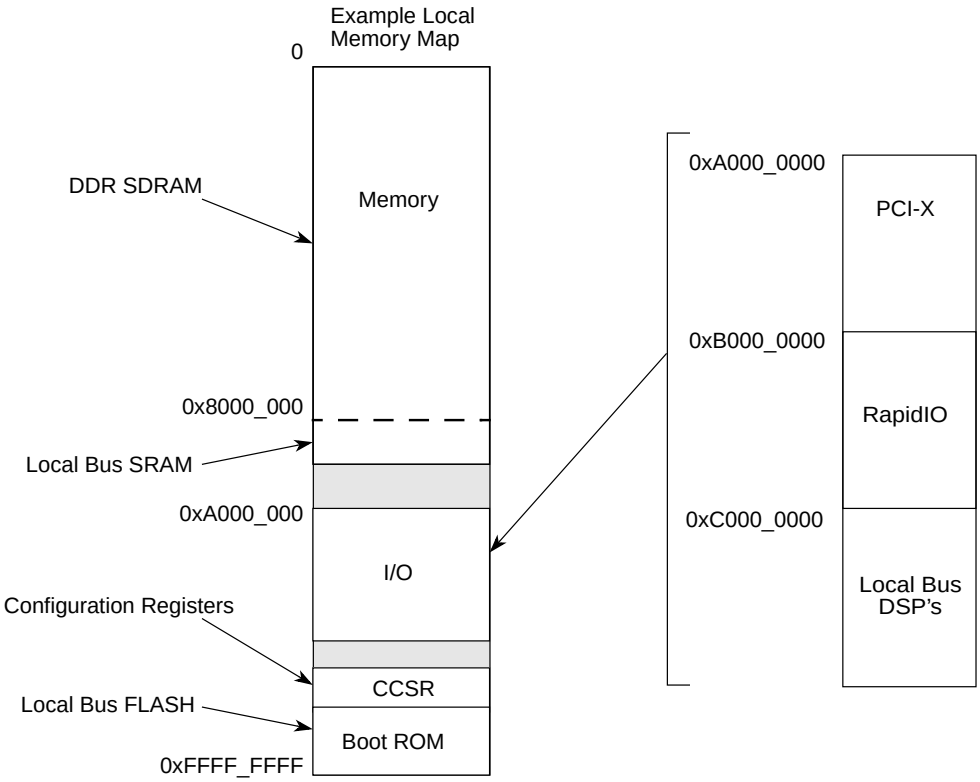


Figure 2-1. Example Local Memory Map

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 MPC8560. 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 MPC8560 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 (snoopable) I/O transactions. Therefore, SRAM windows must never overlap configuration space as defined by CCSRBAR. It is possible to have SRAM windows overlap local access windows, but this is discouraged because processor and snoopable I/O transactions would map to the SRAM while non-snooped I/O transactions would be mapped by the local access windows. Only if all accesses to the SRAM address range are snoopable 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

MPC8560 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)

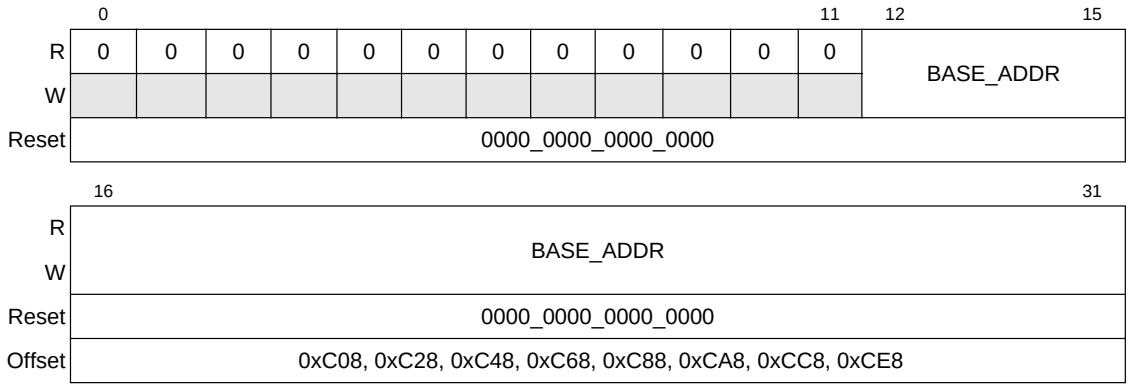


Figure 2-2. Local Access Window *x* Base Address Registers (LAWBAR0–LAWBAR7)

Table 2-5 describes LAWBAR*x* bit settings.

Table 2-5. LAWBAR*x* 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 LAWAR*n*[SIZE].

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

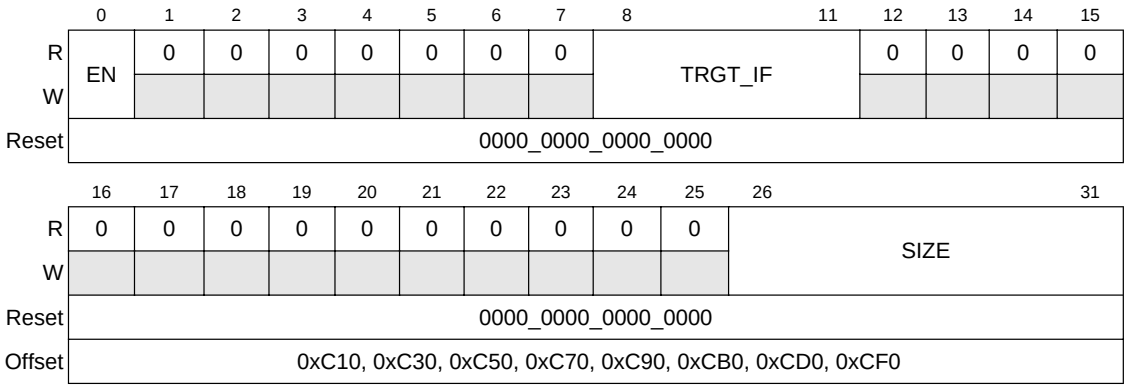


Figure 2-3. Local Access Window *x* Attributes Registers (LAWAR0–LAWAR7)

Table 2-6 describes LAWAR*x* bit settings.

Table 2-6. LAWAR*x* 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. 000000–001010Reserved 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 MPC8560'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 MPC8560, 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 MPC8560. 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 MPC8560 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 MPC8560 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.

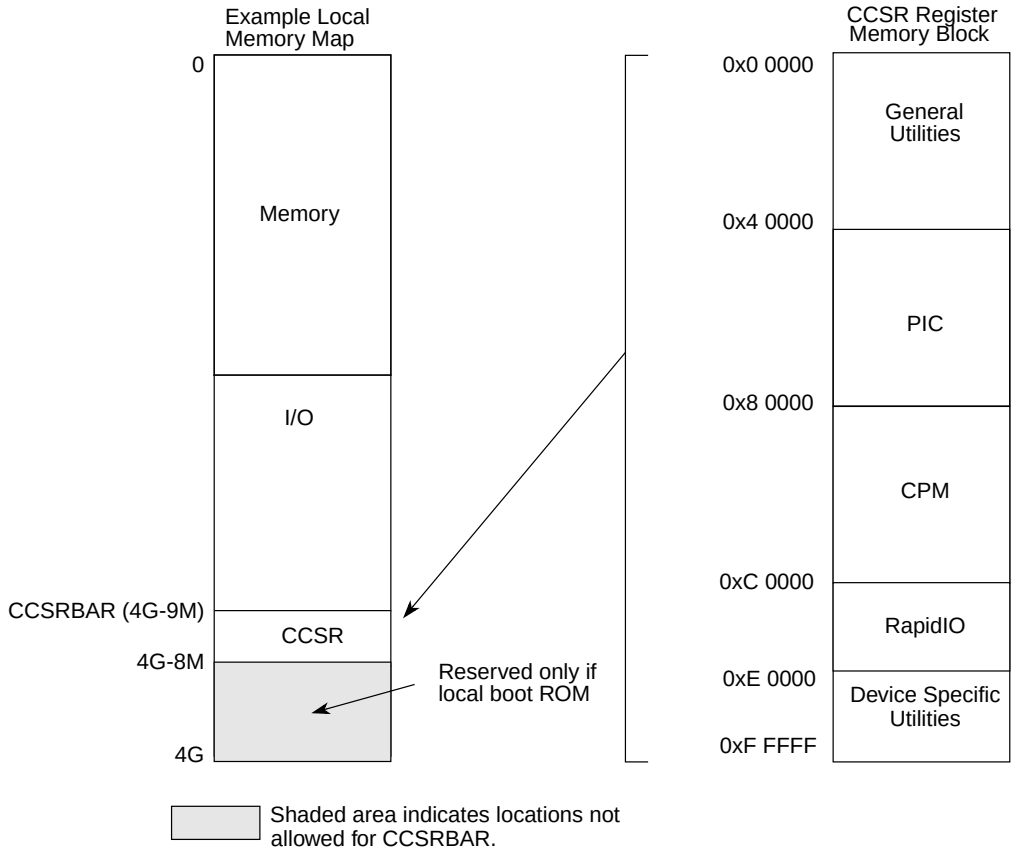


Figure 2-4. Example Top-Level Register Map

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 MPC8560.

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 MPC8560. 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 MPC8560 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	CPM
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.

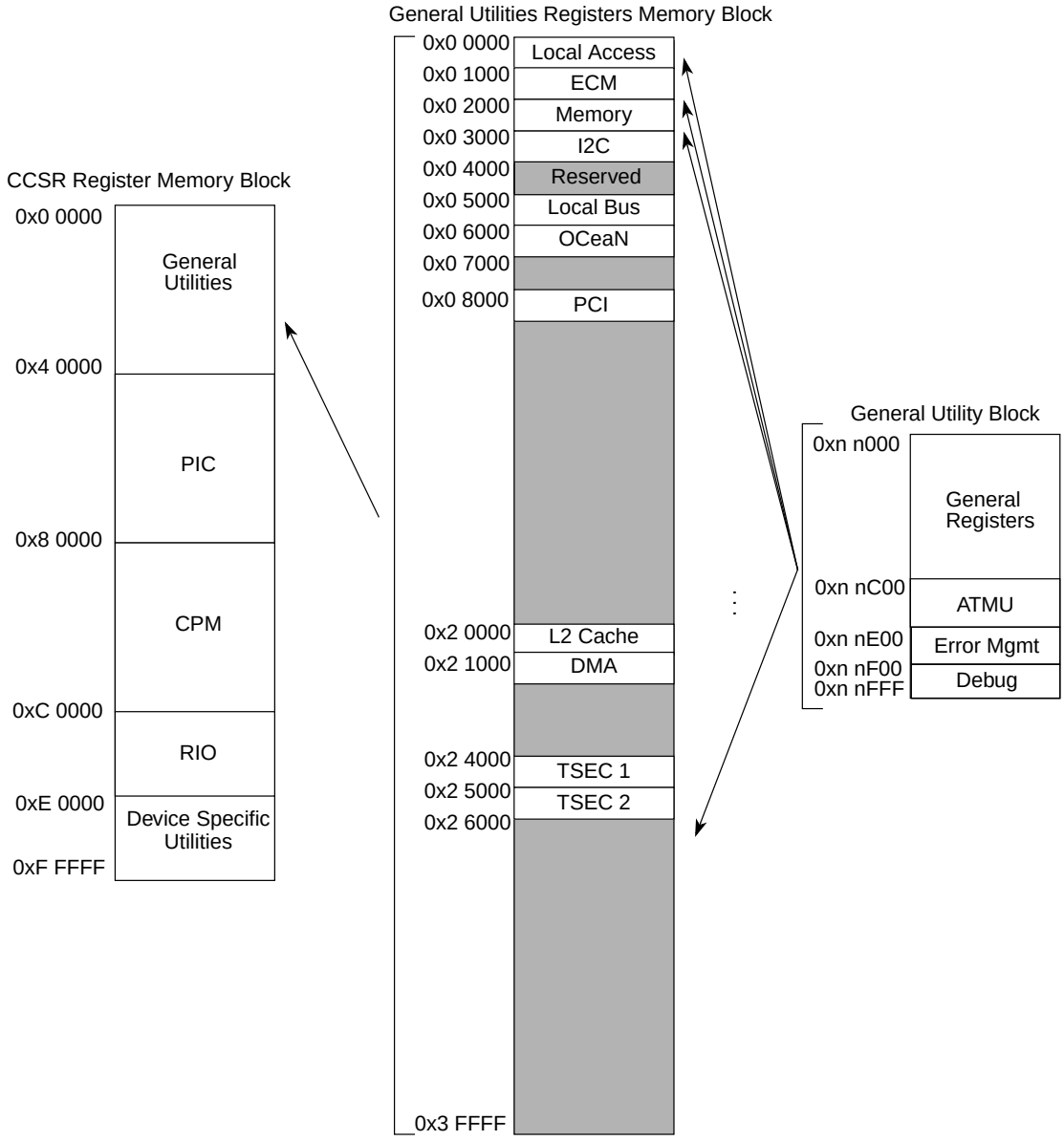


Figure 2-5. General Utilities Registers Mapping to Configuration, Control, and Status Memory Block

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 I^2C register which is 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.

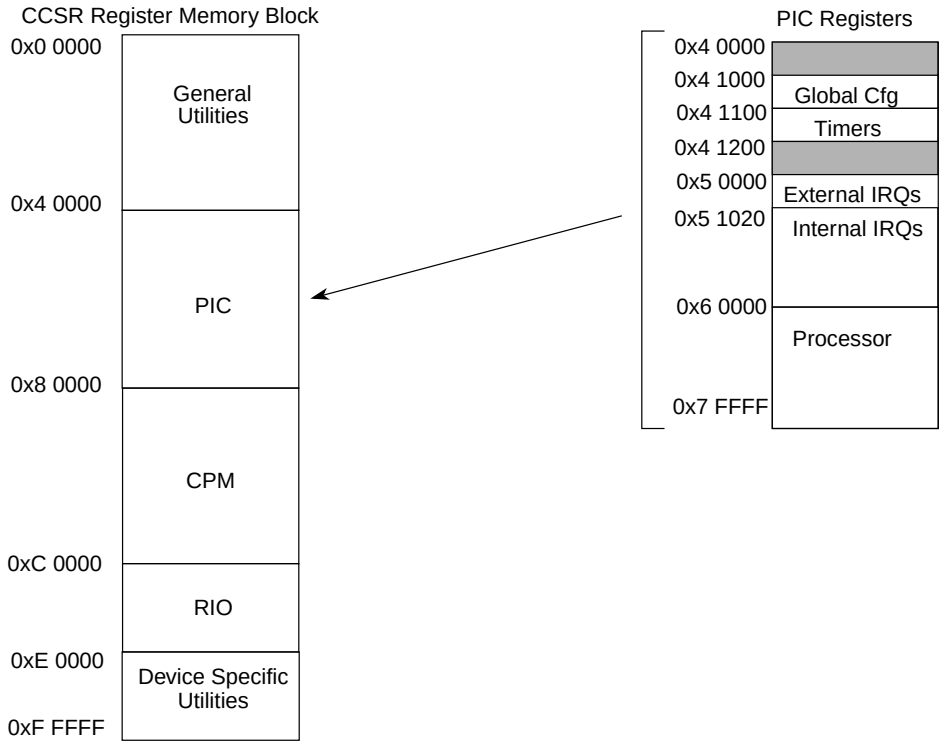


Figure 2-6. PIC Mapping to Configuration, Control, and Status Memory Block

2.3.5 Communications Processor Module and CCSR

The communication processor module (CPM) uses 256 Kbytes of configuration memory. In addition to a 64-Kbyte region for configuration registers, two separate 16-Kbyte parameter RAM regions are defined as well as a 32-Kbyte instruction RAM region.

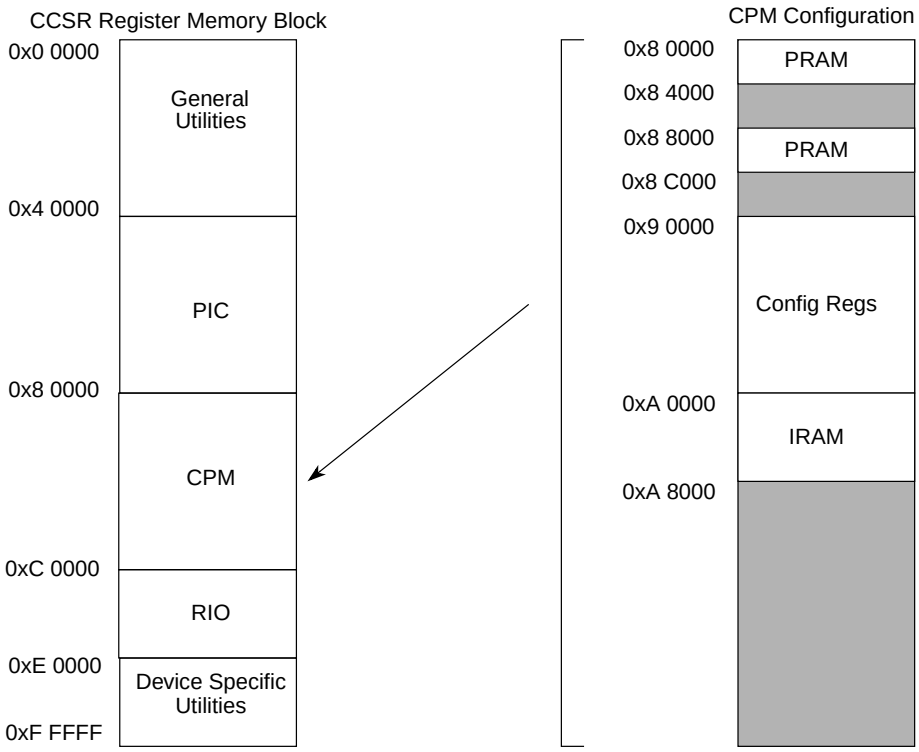


Figure 2-7. CPM Mapping to Configuration, Control, and Status Memory Block

2.3.6 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.

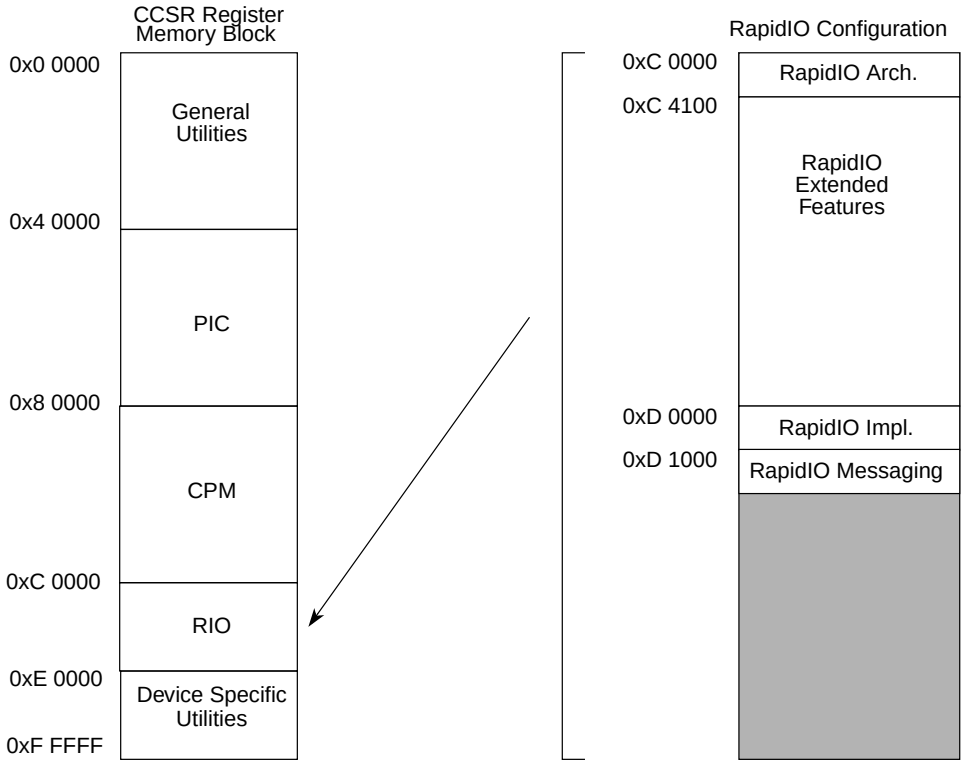


Figure 2-8. RapidIO Mapping to Configuration, Control, and Status Memory Block

2.3.7 Device-Specific Utilities

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

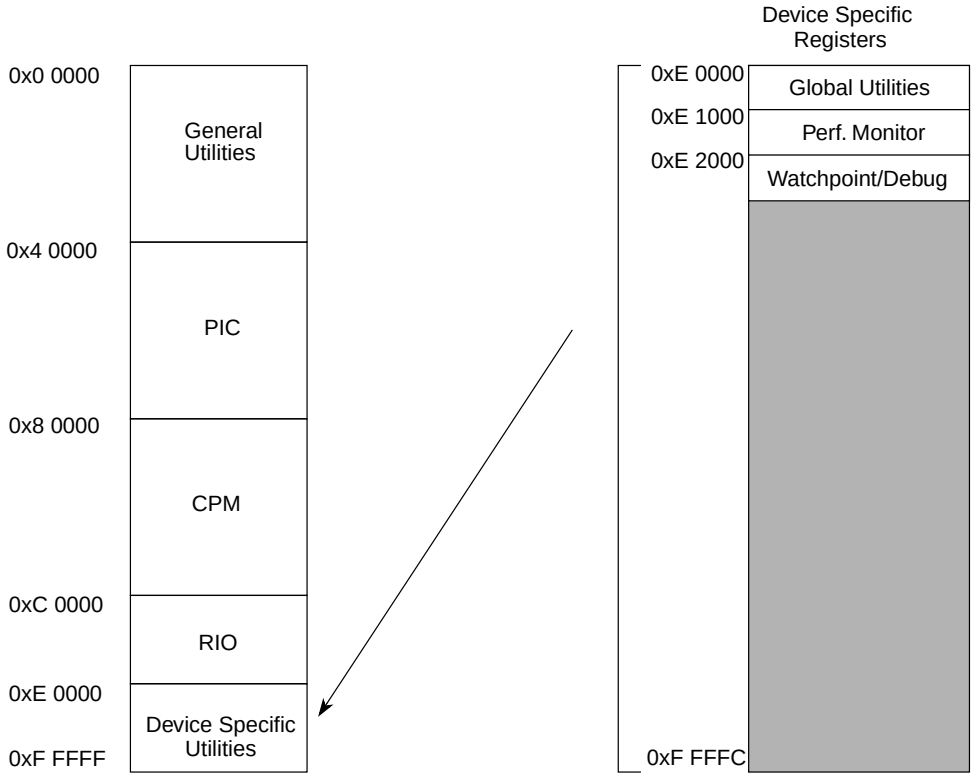


Figure 2-9. Device-Specific Register Mapping to Configuration, Control, and Status Memory Block

2.4 Complete CCSR Map

Table 2-9 lists the MPC8560 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
I^2C				
0x0_3000	I^2C address register (I2CADR)	R/W	0x00	8.3.1.1/8-4
0x0_3004	I^2C frequency divider register (I2CFDR)	R/W	0x00	8.3.1.2/8-5
0x0_3008	I^2C control register (I2CCR)	R/W	0x00	8.3.1.3/8-6
0x0_300C	I^2C status register (I2CSR)	R/W	0x81	8.3.1.4/8-8
0x0_3010	I^2C 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
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)			
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

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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 sestination 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
0x2_6000–0x2_600C	Reserved	R	0x0000_0000	—
PIC Register Address Map—Global Registers				
0x4_0000–0x4_0030	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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	—	—	—
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

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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	$\overline{\text{IRQ_OUT}}$ summary register 0 (IRQSR0)	R	0x0000_0000	10.3.3.1/10-32
0x4_1320	$\overline{\text{IRQ_OUT}}$ 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
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	—	—	—

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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
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

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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
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

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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
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

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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
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

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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	—	—	—
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
CPM Dual-Port RAM				
0x8_0000–0x8_3FFF	Dual-port RAM (DPRAM1)			20.5/-32
0x8_4000–0x8_7FFF	Reserved	—	—	—
0x8_8000–0x8_BFFF	Dual-port RAM (DPRAM2)			20.5/-32
0x8_C000–0x8_FFFF	Reserved	—	—	—
General SIU				
0x9_0050	System bus address error register (SMAER)	R		26.1.3/26-3
0x9_0054	System bus status error register (SMSER)	R		26.1.4/26-3

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x9_0058	System bus event register (SMEVR)	R/W	—	27.1.5/27-4
0x9_005C–0x9_005F	Reserved	—		—
0x9_0060	Local bus address error register (LMAER)	R		27.1.3/27-3
0X9_0064	Local bus status error register (LMSER)	R		27.1.4/27-3
0x9_0068	Local bus event register (LMEVR)	R/W		27.1.5/27-4
0x9_006c–0x9_00FF	Reserved	—	—	—
Interrupt Controller				
0x9_0C00	SIU interrupt configuration register (SICR)	R/W	0x0000_0000	0.1.1.1/4-20
0x9_0C02	Reserved	—	—	—
0x9_0C04	SIU interrupt vector register (SIVEC)	R/W	0x0000_0000	0.1.1.6/4-27
0x9_0C08	SIU interrupt pending register (high) (SIPNR_H)	R/W	0x0000_0000	0.1.1.4/4-24
0x9_0C0C	SIU interrupt pending register (low) (SIPNR_L)	R/W	0x0000_0000	0.1.1.4/4-24
0x9_0C10	Reserved	—	—	—
0x9_0C14	CPM interrupt priority register (high) (SCPRR_H)	R/W	0x0530_9770	0.1.1.3/4-22
0x9_0C18	CPM interrupt priority register (low) (SCPRR_L)	R/W	0x0530_9770	0.1.1.3/4-22
0x9_0C1C	SIU interrupt mask register (high) (SIMR_H)	R/W	0x0000_0000	0.1.1.5/4-26
0x9_0C20	SIU interrupt mask register (low) (SIMR_L)	R/W	0x0000_0000	0.1.1.5/4-26
0x9_0C24	SIU external interrupt control register (SIEXR)	R/W	0x0000_0000	0.1.1.7/4-28
0x9_0C28–0x9_0C7F	Reserved	—	—	—
0x9_0C80	System Clock Control Register (SCCR)	R/W	0x0000_0000	
Input/Output Port				
0x9_0D00	Port A data direction register (PDIRA)	R/W	0x0000_0000	42.2.3/-3
0x9_0D04	Port A pin assignment register (PPARA)	R/W	0x0000_0000	42.2.4/-4
0x9_0D08	Port A special options register (PSORA)	R/W	0x0000_0000	42.2.5/-4
0x9_0D0C	Port A open drain register (PODRA)	R/W	0x0000_0000	42.2.1/-2
0x9_0D10	Port A data register (PDATA)	R/W	0x0000_0000	42.2.2/-2
0x9_0D14–0x9_0D1F	Reserved	—	—	—
0x9_0D20	Port B data direction register (PDIRB)	R/W	0x0000_0000	42.2.3/-3
0x9_0D24	Port B pin assignment register (PPARB)	R/W	0x0000_0000	42.2.4/-4
0x9_0D28	Port B special operation register (PSORB)	R/W	0x0000_0000	42.2.5/-4

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x9_0D2C	Port B open drain register (PODRB)	R/W	0x0000_0000	42.2.1/-2
0x9_0D30	Port B data register (PDATB)	R/W	0x0000_0000	42.2.2/-2
0x9_0D34–0x9_0D3F	Reserved	—	—	—
0x9_0D40	Port C data direction register (PDIRC)	R/W	0x0000_0000	42.2.3/-3
0x9_0D44	Port C pin assignment register (PPARC)	R/W	0x0000_0000	42.2.4/-4
0x9_0D48	Port C special operation register (PSORC)	R/W	0x0000_0000	42.2.5/-4
0x9_0D4C	Port C open drain register (PODRC)	R/W	0x0000_0000	42.2.1/-2
0x9_0D50	Port C data register (PDATC)	R/W	0x0000_0000	42.2.2/-2
0x9_0D54–0x9_0D5F	Reserved	—	—	—
0x9_0D60	Port D data direction register (PDIRD)	R/W	0x0000_0000	42.2.3/-3
0x9_0D64	Port D pin assignment register (PPARD)	R/W	0x0000_0000	42.2.4/-4
0x9_0D68	Port D special operation register (PSORD)	R/W	0x0000_0000	42.2.5/-4
0x9_0D6C	Port D open drain register (PODRD)	R/W	0x0000_0000	42.2.1/-2
0x9_0D70	Port D data register (PDATD)	R/W	0x0000_0000	42.2.2/-2
CPM Timers				
0x9_0D80	Timer 1 and timer 2 global configuration register (TGCR1)	R/W	0x00	25.2.2/-4
0x9_0D81	Reserved	—	—	—
0x9_0D84	Timer 3 and timer 4 global configuration register (TGCR2)	R/W	0x00	25.2.2/-4
0x9_0D85–0x9_0D8F	Reserved	—	—	—
0x9_0D90	Timer 1 mode register (TMR1)	R/W	0x0000	25.2.3/-6
0x9_0D92	Timer 2 mode register (TMR2)	R/W	0x0000	25.2.3/-6
0x9_0D94	Timer 1 reference register (TRR1)	R/W	0x0000	25.2.4/-7
0x9_0D96	Timer 2 reference register (TRR2)	R/W	0x0000	25.2.4/-7
0x9_0D98	Timer 1 capture register (TCR1)	R/W	0x0000	25.2.5/-8
0x9_0D9A	Timer 2 capture register (TCR2)	R/W	0x0000	25.2.5/-8
0x9_0D9C	Timer 1 counter (TCN1)	R/W	0x0000	25.2.6/-8
0x9_0D9E	Timer 2 counter (TCN2)	R/W	0x0000	25.2.6/-8
0x9_0DA0	Timer 3 mode register (TMR3)	R/W	0x0000	25.2.3/-6
0x9_0DA2	Timer 4 mode register (TMR4)	R/W	0x0000	25.2.3/-6

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x9_0DA4	Timer 3 reference register (TRR3)	R/W	0x0000	25.2.4/-7
0x9_0DA6	Timer 4 reference register (TRR4)	R/W	0x0000	25.2.4/-7
0x9_0DA8	Timer 3 capture register (TCR3)	R/W	0x0000	25.2.5/-8
0x9_0DAA	Timer 4 capture register (TCR4)	R/W	0x0000	25.2.5/-8
0x9_0DAC	Timer 3 counter (TCN3)	R/W	0x0000	25.2.6/-8
0x9_0DAE	Timer 4 counter (TCN4)	R/W	0x0000	25.2.6/-8
0x9_0DB0	Timer 1 event register (TER1)	R/W	0x0000	25.2.7/-8
0x9_0DB2	Timer 2 event register (TER2)	R/W	0x0000	25.2.7/-8
0x9_0DB4	Timer 3 event register (TER3)	R/W	0x0000	25.2.7/-8
0x9_0DB6	Timer 4 event register (TER4)	R/W	0x0000	25.2.7/-8
0x9_0DB8–0x9_1017	Reserved	—	—	—
SDMA				
0x9_1018	SDMA status register (SDSR)		0x00	26.1.1/-2
0x9_1019	Reserved	—	—	—
0x9_101C	SDMA mask register (SDMR)	R/W	0x00	26.1.2/-3
0x9_101D	Reserved	—	—	—
FCC1				
0x9_1300	FCC1 general mode register (GFMR1)	R/W	0x0000_0000	34.2/-3
0x9_1304	FCC1 protocol-specific mode register (FPSMR1)	R/W	0x0000_0000	(ATM) 35.13.2/-87 (Ethernet) 37.19.1/-21 (HDLC) 38.6/-8
0x9_1308	FCC1 transmit on demand register (FTODR1)	R/W	0x0000	34.5/-8
0x9_130A	Reserved	—	—	—
0x9_130C	FCC1 data synchronization register (FDSR1)	R/W	0x7E7E	34.4/-7
0x9_130E	Reserved	—	—	—
0x9_1310	FCC1 event register (FCCE1)	R/W	0x0000_0000	(ATM) 35.13.3/-89 (Ethernet) 37.19.2/-23 (HDLC) 38.9/-14
0x9_1312	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x9_1314	FCC1 mask register (FCCM1)	R/W	0x0000_0000	(ATM) 35.13.3/-89 (Ethernet) 37.19.2/-23 (HDLC) 38.9/-14
0x9_1316	Reserved	—	—	—
0x9_1318	FCC1 status register (FCCS1)	R	0x00	38.10/-16 (HDLC)
0x9_1319	Reserved	—	—	—
0x9_131C	FCC1 transmit internal rate registers for PHY0 (FTIRR1_PHY0)	R/W	0x00	35.13.4/-91 (ATM)
0x9_131D	FCC1 transmit internal rate registers for PHY1 (FTIRR1_PHY1)	R/W	0x00	
0x9_131E	FCC1 transmit internal rate registers for PHY2 (FTIRR1_PHY2)	R/W	0x00	
0x9_131F	FCC1 transmit internal rate registers for PHY3 (FTIRR1_PHY3)	R/W	0x00	
FCC2				
0x9_1320	FCC2 general mode register (GFMR2)	R/W	0x0000_0000	34.2/-3
0x9_1324	FCC2 protocol-specific mode register (FPSMR2)	R/W	0x0000_0000	(ATM) 35.13.2/-87 (Ethernet) 37.19.1/-21 (HDLC) 38.6/-8
0x9_1328	FCC2 transmit on-demand register (FTODR2)	R/W	0x0000	34.5/-8
0x9_132A	Reserved	—	—	—
0x9_132C	FCC2 data synchronization register (FDSR2)	R/W	0x7E7E	34.4/-7
0x9_132E	Reserved	—	—	—
0x9_1330	FCC2 event register (FCCE2)	R/W	0x0000_0000	(ATM) 35.13.3/-89 (Ethernet) 37.19.2/-23 (HDLC) 38.9/-14
0x9_1332	Reserved	—	—	—
0x9_1334	FCC2 mask register (FCCM2)	R/W	0x0000_0000	(ATM) 35.13.3/-89 (Ethernet) 37.19.2/-23 (HDLC) 38.9/-14

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x9_1336	Reserved	—	—	—
0x9_1338	FCC2 status register (FCCS2)	R	0x00	38.10/-16 (HDLC)
0x9_1339	Reserved	—	0x00	—
0x9_133C	FCC2 transmit internal rate registers for PHY0 (FTIRR2_PHY0)	R/W	0x00	35.13.4/-91 (ATM)
0x9_133D	FCC2 transmit internal rate registers for PHY1 (FTIRR2_PHY1)	R/W	0x00	
0x9_133E	FCC2 transmit internal rate registers for PHY2 (FTIRR2_PHY2)	R/W	0x00	
0x9_133F	FCC2 transmit internal rate registers for PHY3 (FTIRR2_PHY3)	R/W	0x00	
FCC3				
0x9_1340	FCC3 general mode register (GFMR3)	R/W	0x0000_0000	34.2/-3
0x9_1344	FCC3 protocol-specific mode register (FPSMR3)	R/W	0x0000_0000	(ATM) 35.13.2/-87 (Ethernet) 37.19.1/-21 (HDLC) 38.6/-8
0x9_1348	FCC3 transmit on-demand register (FTODR3)	R/W	0x0000	34.5/-8
0x9_134A	Reserved	—	—	—
0x9_134C	FCC3 data synchronization register (FDSR3)	R/W	0x7E7E	34.4/-7
0x9_134E	Reserved	—	—	—
0x9_1350	FCC3 event register (FCCE3)	R/W	0x0000_0000	(ATM) 35.13.3/-89 (Ethernet) 37.19.2/-23 (HDLC) 38.9/-14
0x9_1352	Reserved	—	—	—
0x9_1354	FCC3 mask register (FCCM3)	R/W	0x0000_0000	(ATM) 35.13.3/-89 (Ethernet) 37.19.2/-23 (HDLC) 38.9/-14
0x9_1356	Reserved	—	—	—
0x9_1358	FCC3 status register (FCCS3)	R	0x00	38.10/-16 (HDLC)
0x9_1359–0x9_137F	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
FCC1 (continued)				
0x9_1380	FCC1 internal rate port enable register (FIRPER1)	R/W	0x0000_0000	
0x9_1384	FCC1 internal rate event register (FIRER1)	R/W	0x0000_0000	
0x9_1388	FCC1 internal rate selection register:HI (FIRSR1_HI)	R/W	0x0000_0000	
0x9_138C	FCC1 internal rate selection register:LO (FIRSR1_LO)	R/W	0x0000_0000	
0x9_1390	General FCC1 expansion mode register (GFEMR1)	R/W	0x00	
0x9_1391–0x9_139F	Reserved	—	—	—
FCC2 (continued)				
0x9_13A0	FCC2 internal rate port enable register (FIRPER2)	R/W	0x0000_0000	
0x9_13A4	FCC2 internal rate event register (FIRER2)	R/W	0x0000_0000	
0x9_13A8	FCC2 internal rate selection register:HI (FIRSR2_HI)	R/W	0x0000_0000	
0x9_13AC	FCC2 internal rate selection register:LO (FIRSR2_LO)	R/W	0x0000_0000	
0x9_13B0	General FCC2 expansion mode register (GFEMR2)	R/W	0x00	
0x9_13B1–0x9_13CF	Reserved	—	—	—
FCC3 (continued)				
0x9_13D0	General FCC3 expansion mode register (GFEMR3)	R/W	0x00	
0x9_13D1–0x9_13FF	Reserved	—	—	—
TC Layer 1				
0x9_1400	TC1 mode register (TCMODE1)			36.11.1.1/-8
0x9_1402	TC1 cell delineation state machine register (CDSMR1)			36.11.1.2/-10
0x9_1404	TC1 event register (TCER1)			36.11.1.3/-10
0x9_1406	TC1 received cells counter (TC_RCC1)			36.11.3.1/-14
0x9_1408	TC1 mask register (TCMR1)			36.11.1.4/-11
0x9_140A	TC1 filtered cells counter (TC_FCC1)			36.11.3.6/-14

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x9_140C	TC1 corrected cells counter (TC_CCC1)			36.11.3.4/-14
0x9_140E	TC1 idle cells counter (TC_ICC1)			36.11.3.5/-14
0x9_1410	TC1 transmitted cells counter (TC_TCC1)			36.11.3.2/-14
0x9_1412	TC1 error cells counter (TC_ECC1)			36.11.3.3/-14
0x9_1414	Reserved	—	—	—
TC Layer 2				
0x9_1420	TC2 mode register (TCMODE2)			36.11.1.1/-8
0x9_1422	TC2 cell delineation state machine register (CDSMR2)			36.11.1.2/-10
0x9_1424	TC2 event register (TCER2)			36.11.1.3/-10
0x9_1426	TC2 received cells counter (TC_RCC2)			36.11.3.1/-14
0x9_1428	TC2 mask register (TCMR2)			36.11.1.4/-11
0x9_142A	TC2 filtered cells counter (TC_FCC2)			36.11.3.6/-14
0x9_142C	TC2 corrected cells counter (TC_CCC2)			36.11.3.4/-14
0x9_142E	TC2 idle cells counter (TC_ICC2)			36.11.3.5/-14
0x9_1430	TC2 transmitted cells counter (TC_TCC2)			36.11.3.2/-14
0x9_1432	TC2 error cells counter (TC_ECC2)			36.11.3.3/-14
0x9_1434	Reserved	—	—	—
TC Layer 3				
0x9_1440	TC3 mode register (TCMODE3)			36.11.1.1/-8
0x9_1442	TC3 cell delineation state machine register (CDSMR3)			36.11.1.2/-10
0x9_1444	TC3 event register (TCER3)			36.11.1.3/-10
0x9_1446	TC3 received cells counter (TC_RCC3)			36.11.3.1/-14
0x9_1448	TC3 mask register (TCMR3)			36.11.1.4/-11
0x9_144A	TC3 filtered cells counter (TC_FCC3)			36.11.3.6/-14
0x9_144C	TC3 corrected cells counter (TC_CCC3)			36.11.3.4/-14
0x9_144E	TC3 idle cells counter (TC_ICC3)			36.11.3.5/-14
0x9_1450	TC3 transmitted cells counter (TC_TCC3)			36.11.3.2/-14
0x9_1452	TC3 error cells counter (TC_ECC3)			36.11.3.3/-14
0x9_1454	Reserved	—	—	—
TC Layer 4				
0x9_1460	TC4 mode register (TCMODE4)			36.11.1.1/-8

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x9_1462	TC4 cell delineation state machine register (CDSMR4)			36.11.1.2/-10
0x9_1464	TC4 event register (TCER4)			36.11.1.3/-10
0x9_1466	TC4 received cells counter (TC_RCC4)			36.11.3.1/-14
0x9_1468	TC4 mask register (TCMR4)			36.11.1.4/-11
0x9_146A	TC4 filtered cells counter (TC_FCC4)			36.11.3.6/-14
0x9_146C	TC4 corrected cells counter (TC_CCC4)			36.11.3.4/-14
0x9_146E	TC4 idle cells counter (TC_ICC4)			36.11.3.5/-14
0x9_1470	TC4 transmitted cells counter (TC_TCC4)			36.11.3.2/-14
0x9_1472	TC4 error cells counter (TC_ECC4)			36.11.3.3/-14
0x9_1474	Reserved	—	—	—
TC Layer 5				
0x9_1480	TC5 mode register (TCMODE5)			36.11.1.1/-8
0x9_1482	TC5 cell delineation state machine register (CDSMR5)			36.11.1.2/-10
0x9_1484	TC5 event register (TCER5)			36.11.1.3/-10
0x9_1486	TC5 received cells counter (TC_RCC5)			36.11.3.1/-14
0x9_1488	TC5 mask register (TCMR5)			36.11.1.4/-11
0x9_148A	TC5 filtered cells counter (TC_FCC5)			36.11.3.6/-14
0x9_148C	TC5 corrected cells counter (TC_CCC5)			36.11.3.4/-14
0x9_148E	TC5 idle cells counter (TC_ICC5)			36.11.3.5/-14
0x9_1490	TC5 transmitted cells counter (TC_TCC5)			36.11.3.2/-14
0x9_1492	TC5 error cells counter (TC_ECC5)			36.11.3.3/-14
0x9_1494	Reserved	—	—	—
TC Layer 6				
0x9_14A0	TC6 mode register (TCMODE6)			36.11.1.1/-8
0x9_14A2	TC6 cell delineation state machine register (CDSMR6)			36.11.1.2/-10
0x9_14A4	TC6 event register (TCER6)			36.11.1.3/-10
0x9_14A6	TC6 received cells counter (TC_RCC6)			36.11.3.1/-14
0x9_14A8	TC6 mask register (TCMR6)			36.11.1.4/-11
0x9_14AA	TC6 filtered cells counter (TC_FCC6)			36.11.3.6/-14
0x9_14AC	TC6 corrected cells counter (TC_CCC6)			36.11.3.4/-14

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x9_14AE	TC6 idle cells counter (TC_ICC6)			36.11.3.5/-14
0x9_14B0	TC6 transmitted cells counter (TC_TCC6)			36.11.3.2/-14
0x9_14B2	TC6 error cells counter (TC_ECC6)			36.11.3.3/-14
0x9_14B4	Reserved	—	—	—
TC Layer 7				
0x9_14C0	TC7 mode register (TCMODE7)			36.11.1.1/-8
0x9_14C2	TC7 cell delineation state machine register (CDSMR7)			36.11.1.2/-10
0x9_14C4	TC7 event register (TCER7)			36.11.1.3/-10
0x9_14C6	TC7 received cells counter (TC_RCC7)			36.11.3.1/-14
0x9_14C8	TC7 mask register (TCMR7)			36.11.1.4/-11
0x9_14CA	TC7 filtered cells counter (TC_FCC7)			36.11.3.6/-14
0x9_14CC	TC7 corrected cells counter (TC_CCC7)			36.11.3.4/-14
0x9_14CE	TC7 idle cells counter (TC_ICC7)			36.11.3.5/-14
0x9_14D0	TC7 transmitted cells counter (TC_TCC7)			36.11.3.2/-14
0x9_14D2	TC7 error cells counter (TC_ECC7)			36.11.3.3/-14
0x9_14D4	Reserved	—	—	—
TC Layer 8				
0x9_14E0	TC8 mode register (TCMODE8)			36.11.1.1/-8
0x9_14E2	TC8 cell delineation state machine register (CDSMR8)			36.11.1.2/-10
0x9_14E4	TC8 event register (TCER8)			36.11.1.3/-10
0x9_14E6	TC8 received cells counter (TC_RCC8)			36.11.3.1/-14
0x9_14E8	TC8 mask register (TCMR8)			36.11.1.4/-11
0x9_14EA	TC8 filtered cells counter (TC_FCC8)			36.11.3.6/-14
0x9_14EC	TC8 corrected cells counter (TC_CCC8)			36.11.3.4/-14
0x9_14EE	TC8 idle cells counter (TC_ICC8)			36.11.3.5/-14
0x9_14F0	TC8 transmitted cells counter (TC_TCC8)			36.11.3.2/-14
0x9_14F2	TC8 error cells counter (TC_ECC8)			36.11.3.3/-14
0x9_14F4	Reserved	—	—	—
TC Layer—General				
0x9_1500	TC general status register (TCGSR)			36.11.2.2/-13
0x9_1502	TC general event register (TCGER)			36.11.2.1/-12

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
BRGs 5–8				
0x9_15F0	BRG5 configuration register (BRGC5)	R/W	0x0000_0000	24.2/-4
0x9_15F4	BRG6 configuration register (BRGC6)	R/W	0x0000_0000	
0x9_15F8	BRG7 configuration register (BRGC7)	R/W	0x0000_0000	
0x9_15FC	BRG8 configuration register (BRGC8)	R/W	0x0000_0000	
0x9_1600–0x9_185F	Reserved	—	—	—
I^2C				
0x9_1860	I^2C mode register (I2MOD)	R/W	0x00	41.4.1/-6
0x9_1861	Reserved	—	—	—
0x9_1864	I^2C address register (I2ADD)	R/W	0x00	41.4.2/-7
0x9_1865	Reserved	—	—	—
0x9_1868	I^2C BRG register (I2BRG)	R/W	0x00	41.4.3/-7
0x9_1869	Reserved	—	—	—
0x9_186C	I^2C command register (I2COM)	R/W	0x00	41.4.5/-8
0x9_186D	Reserved	—	—	—
0x9_1870	I^2C event register (I2CER)	R/W	0x00	41.4.4/-8
0x9_1871	Reserved	—	—	—
0x9_1874	I^2C mask register (I2CMR)	R/W	0x00	41.4.4/-8
0x9_1875–0x9_19BF	Reserved	—	—	—
Communications Processor				
0x9_19C0	Communications processor command register (CPCR)	R/W	0x0000_0000	20.4.1/-28
0x9_19C4	CP configuration register (RCCR)	R/W	0x0000_0000	20.3.6/-25
0x9_19C8–0x9_19D5	Reserved	—	—	—
0x9_19D6	CP timers event register (RTER)	R/W	0x0000	20.6.4/-40
0x9_19DA	CP timers mask register (RTMR)	R/W	0x0000	
0x9_19DC	CP time-stamp timer control register (RTSCR)	R/W	0x0000	20.3.7/-26
0x9_19DE	Reserved	—	—	—
0x9_19E0	CP time-stamp register (RTSR)	R/W	0x0000_0000	20.3.8/-27

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
BRGs 1–4				
0x9_19F0	BRG1 configuration register (BRGC1)	R/W	0x0000_0000	24.2/-4
0x9_19F4	BRG2 configuration register (BRGC2)	R/W	0x0000_0000	
0x9_19F8	BRG3 configuration register (BRGC3)	R/W	0x0000_0000	
0x9_19FC	BRG4 configuration register (BRGC4)	R/W	0x0000_0000	
SCC1				
0x9_1A00	SCC1 general mode register (GSMR_L1)	R/W	0x0000_0000	27.2/-3
0x9_1A04	SCC1 general mode register (GSMR_H1)	R/W	0x0000_0000	
0x9_1A08	SCC1 protocol-specific mode register (PSMR1)	R/W	0x0000	27.2.1/-9 28.16/-14 (UART) 29.8/-7 (HDLC) 30.11/-10 (BISYNC) 31.9/-9 (Transparent)
0x9_1A0A	Reserved	—	—	—
0x9_1A0C	SCC1 transmit-on-demand register (TODR1)	R/W	0x0000	27.2.3/-10
0x9_1A0E	SCC1 data synchronization register (DSR1)	R/W	0x7E7E	27.2.2/-9
0x9_1A10	SCC1 event register (SCCE1)	R/W		28.19/-20 (UART) 29.11/-13 (HDLC) 30.14/-15 (BISYNC) 31.12/-12 (Transparent)
0x9_1A14	SCC1 mask register (SCCM1)	R/W	0x0000	
0x9_1A16	Reserved	—	—	—
0x9_1A17	SCC1 status register (SCCS1)	R/W	0x00	28.20/-22 (UART) 29.12/-15 (HDLC) 30.15/-16 (BISYNC) 31.13/-13 (Transparent)
0x9_1A18– 0x9_1A1F	Reserved	—	—	—
SCC2				
0x9_1A20	SCC2 general mode register (low) (GSMR_L2)	R/W	0x0000_0000	27.2/-3
0x9_1A24	SCC2 general mode register (high) (GSMR_H2)	R/W	0x0000_0000	

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x9_1A28	SCC2 protocol-specific mode register (PSMR2)	R/W	0x0000	27.2.1/-9 28.16/-14 (UART) 29.8/-7 (HDLC) 30.11/-10 (BISYNC) 31.9/-9 (Transparent)
0x9_1A2A	Reserved	—	—	—
0x9_1A2C	SCC2 transmit-on-demand register (TODR2)	R/W	0x0000	27.2.3/-10
0x9_1A2E	SCC2 data synchronization register (DSR2)	R/W	0x7E7E	27.2.2/-9
0x9_1A30	SCC2 event register (SCCE2)	R/W		28.19/-20 (UART) 29.11/-13 (HDLC) 30.14/-15 (BISYNC) 31.12/-12 (Transparent)
0x9_1A34	SCC2 mask register (SCCM2)	R/W	0x0000	
0x9_1A36	Reserved	—	—	—
0x9_1A37	SCC2 status register (SCCS2)	R/W	0x00	28.20/-22 (UART) 29.12/-15 (HDLC) 30.15/-16 (BISYNC) 31.13/-13 (Transparent)
0x9_1A38– 0x9_1A3F	Reserved	—	—	—
SCC3				
0x9_1A40	SCC3 general mode register (GSMR_L3)	R/W	0x0000_0000	27.2/-3
0x9_1A44	SCC3 general mode register (GSMR_H3)	R/W	0x0000_0000	
0x9_1A48	SCC3 protocol-specific mode register (PSMR3)	R/W	0x0000	27.2.1/-9 28.16/-14 (UART) 29.8/-7 (HDLC) 30.11/-10 (BISYNC) 31.9/-9 (Transparent)
0x9_1A4A	Reserved	—	—	—
0x9_1A4C	SCC3 transmit on demand register (TODR3)	R/W	0x0000	27.2.3/-10
0x9_1A4E	SCC3 data synchronization register (DSR3)	R/W	0x7E7E	27.2.2/-9

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x9_1A50	SCC3 event register (SCCE3)	R/W		28.19/-20 (UART) 29.11/-13 (HDLC) 30.14/-15 (BISYNC) 31.12/-12 (Transparent)
0x9_1A54	SCC3 mask register (SCCM3)	R/W	0x0000	
0x9_1A56	Reserved	—	—	—
0x9_1A57	SCC3 status register (SCCS3)	R/W	0x00	28.20/-22 (UART) 29.12/-15 (HDLC) 30.15/-16 (BISYNC) 31.13/-13 (Transparent)
0x9_1A58–0x9_1A5F	Reserved	—	—	—
SCC4				
0x9_1A60	SCC4 general mode register (GSMR_L4)	R/W	0x0000_0000	27.2/-3
0x9_1A64	SCC4 general mode register (GSMR_H4)	R/W	0x0000_0000	
0x9_1A68	SCC4 protocol-specific mode register (PSMR4)	R/W	0x0000	27.2.1/-9 28.16/-14 (UART) 29.8/-7 (HDLC) 30.11/-10 (BISYNC) 31.9/-9 (Transparent)
0x9_1A6A	Reserved	—	—	—
0x9_1A6C	SCC4 transmit on-demand register (TODR4)	R/W	0x0000	27.2.3/-10
0x9_1A6E	SCC4 data synchronization register (DSR4)	R/W	0x7E7E	27.2.2/-9
0x9_1A70	SCC4 event register (SCCE4)	R/W		28.19/-20 (UART) 29.11/-13 (HDLC) 30.14/-15 (BISYNC) 31.12/-12 (Transparent)
0x9_1A74	SCC4 mask register (SCCM4)	R/W	0x0000	
0x9_1A76	Reserved	—	—	—
0x9_1A77	SCC4 status register (SCCS4)	—	—	28.20/-22 (UART) 29.12/-15 (HDLC) 30.15/-16 (BISYNC) 31.13/-13 (Transparent)
0x9_1A78–0x9_1A7F	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
SPI				
0x9_1AA0	SPI mode register (SPMODE)	R/W	0x0000	40.4.1/-7
0x9_1AA2	Reserved	—	—	—
0x9_1AA6	SPI event register (SPIE)	R/W	0x00	40.4.2/-10
0x9_1AA7	Reserved	—	—	—
0x9_1AAA	SPI mask register (SPIM)	R/W	0x00	40.4.2/-10
0x9_1AAB	Reserved	—	—	—
0x9_1AAD	SPI command register (SPCOM)	W	0x00	40.4.3/-11
0x9_1AA7–0x9_1AFF	Reserved	—	—	—
CPM Mux				
0x9_1B00	CPM mux SI1 clock route register (CMXSI1CR)	R/W	0x00	23.4.2/-12
0x9_1B02	CPM mux SI2 clock route register (CMXSI2CR)	R/W	0x00	23.4.3/-12
0x9_1B03	Reserved	—	—	—
0x9_1B04	CPM mux FCC clock route register (CMXFCR)	R/W	0x0000_0000	23.4.4/-13
0x9_1B08	CPM mux SCC clock route register (CMXSCR)	R/W	0x0000_0000	23.4.5/-16
0x9_1B0C	Reserved	—	—	—
0x9_1B0E	CPM mux UTOPIA address register (CMXUAR)	R/W	0x0000	23.4.1/-8
0x9_1B10–0x9_1B1F	Reserved	—	—	—
SI1 Registers				
0x9_1B20	SI1 TDMA1 mode register (SI1AMR)	R/W	0x0000	22.5.2/-18
0x9_1B22	SI1 TDMB1 mode register (SI1BMR)	R/W	0x0000	
0x9_1B24	SI1 TDMC1 mode register (SI1CMR)	R/W	0x0000	
0x9_1B26	SI1 TDMD1 mode register (SI1DMR)	R/W	0x0000	
0x9_1B28	SI1 global mode register (SI1GMR)	R/W	0x00	22.5.1/-18
0x9_1B29	Reserved	—	—	—
0x9_1B2A	SI1 command register (SI1CMDR)	R/W	0x00	22.5.4/-25
0x9_1B2B	Reserved	—	—	—
0x9_1B2C	SI1 status register (SI1STR)	R/W	0x00	22.5.5/-26
0x9_1B2D	Reserved	—	—	—
0x9_1B2E	SI1 RAM shadow address register (SI1RSR)	R/W	0x0000	22.5.3/-24
MCC1 Registers				

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x9_1B30	MCC1 event register (MCCE1)	R/W	0x0000	33.10.1/-18
0x9_1B32	Reserved	—	—	—
0x9_1B34	MCC1 mask register (MCCM1)	R/W	0x0000	33.10.1/-18
0x9_1B36	Reserved	—	—	—
0x9_1B38	MCC1 configuration register (MCCF1)	R/W	0x00	33.8/-15
0x9_1B39–0x9_1B3F	Reserved	—	—	—
SI2 Registers				
0x9_1B40	SI2 TDMA2 mode register (SI2AMR)	R/W	0x0000	22.5.2/-18
0x9_1B42	SI2 TDMB2 mode register (SI2BMR)	R/W	0x0000	
0x9_1B44	SI2 TDMC2 mode register (SI2CMR)	R/W	0x0000	
0x9_1B46	SI2 TDMD2 mode register (SI2DMR)	R/W	0x0000	
0x9_1B48	SI2 global mode register (SI2GMR)	R/W	0x00	22.5.1/-18
0x9_1B49	Reserved	—	—	—
0x9_1B4A	SI2 command register (SI2CMDR)	R/W	0x00	22.5.4/-25
0x9_1B4B	Reserved	—	—	—
0x9_1B4C	SI2 status register (SI2STR)	R/W	0x00	22.5.5/-26
0x9_1B4D	Reserved	—	—	—
0x9_1B4E	SI2 RAM shadow address register (SI2RSR)	R/W	0x0000	22.5.3/-24
MCC2 Registers				
0x9_1B50	MCC2 event register (MCCE2)	R/W	0x0000	33.10.1/-18
0x9_1B52	Reserved	—	—	—
0x9_1B54	MCC2 mask register (MCCM2)	R/W	0x0000	33.10.1/-18
0x9_1B56	Reserved	—	—	—
0x9_1B58	MCC2 configuration register (MCCF2)	R/W	0x00	33.8/-15
0x9_1B59–0x9_1FFF	Reserved	—	—	—
SI1 RAM				
0x9_2000–0x9_21FF	SI 1 transmit routing RAM (SI1TxRAM)			22.4.3/-10
0x9_2200–0x9_23FF	Reserved	—	—	—
0x9_2400–0x9_25FF	SI 1 receive routing RAM (SI1RxRAM)			22.4.3/-10

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x9_2600–0x9_27FF	Reserved	—	—	—
SI2 RAM				
0x9_2800–0x9_29FF	SI 2 transmit routing RAM (SI2TxRAM			22.4.3/-10
0x9_2A00–0x9_2BFF	Reserved	—	—	—
0x9_2C00–0x9_2DFF	SI 2 receive routing RAM (SI2RxRAM)			22.4.3/-10
0x9_2E00–0x9_3FFF	Reserved	—	—	—
CPM Registers				
0xA_0000–0xA_7FFF	Dual-port RAM (instruction RAM only)			
0xA_8000–0xB_FFFF	Reserved	—	—	—
RapidIO Registers				
Architectural Registers				
0xC_0000	Device identity capability register (DIDCAR)	R	0x0003_0002	17.3.2.1/17-14
0xC_0004	Device information capability register (DICAR)	R	0x8060_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

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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
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

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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
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

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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[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

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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				
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 pointeraddress 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

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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
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	0xnnnn_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	MPC8560 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 (DDRDLLCR)	R/W	0x0000_0000	18.3.1.18/18-29

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
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
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				

Table 2-9. Memory Map (continued)

Local Memory Offset (Hex)	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
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 MPC8560'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

MPC8560 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
- Local bus interface signals
- DMA interface signals
- PIC interface signals
- I^2C interface signals
- CPM interface signals
- System control, power management, and debug signals
- Test, JTAG, and configuration signals
- Clock signals

Figure 3-1 illustrates the external signals of the MPC8560, showing how the signals are grouped. Refer to the *MPC8560 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.

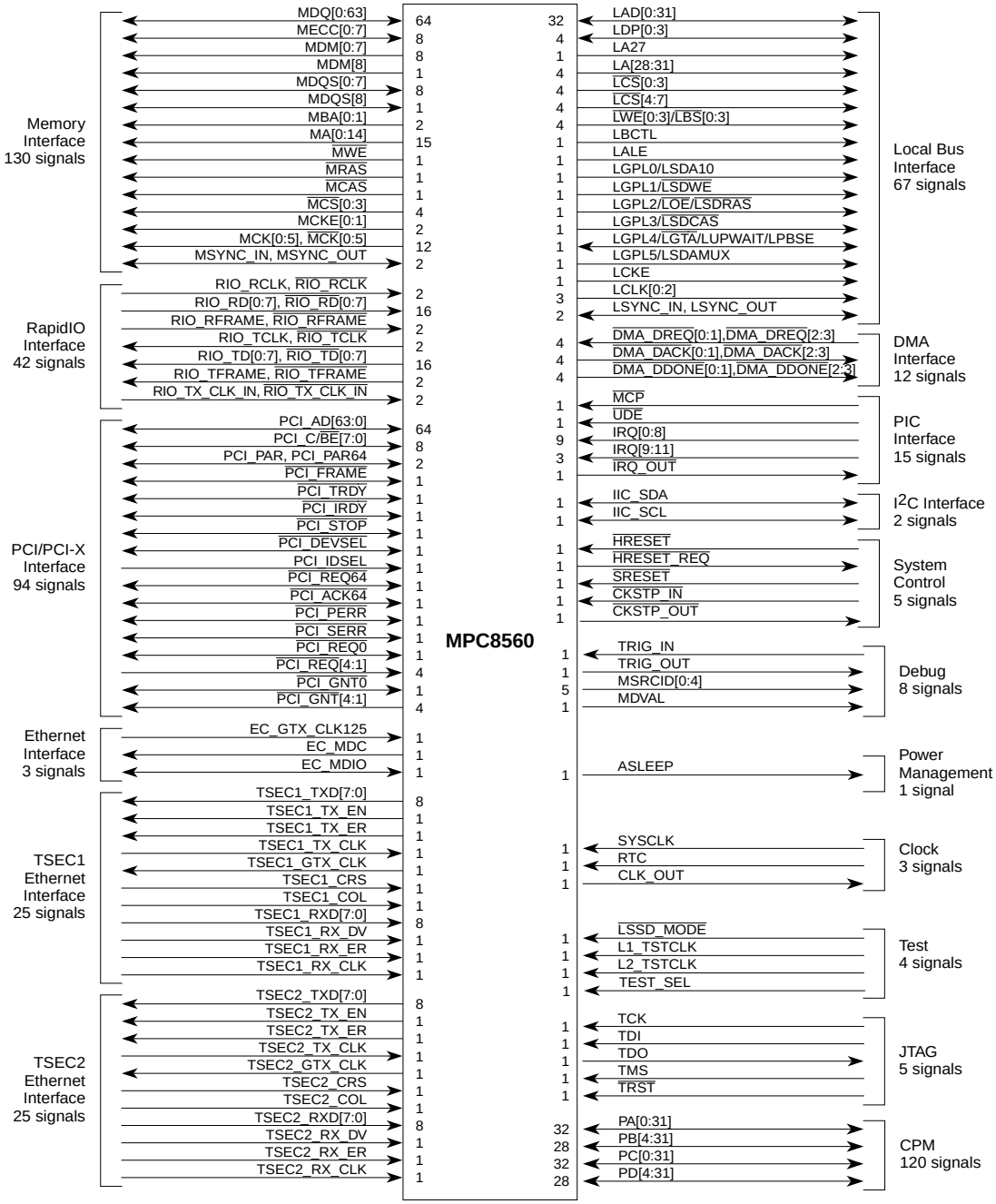


Figure 3-1. MPC8560 Signal Groupings

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. MPC8560 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], $\overline{\text{MCK}}$[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	16-4/16-8
RIO_RD[0:7], $\overline{\text{RIO_RD}}$[0:7]	RapidIO Receive Data	RapidIO	—	16	I	16-4/16-8
RIO_RFRAME, $\overline{\text{RIO_RFRAME}}$	RapidIO Receive Frame	RapidIO	—	2	I	16-4/16-8
RIO_TCLK, $\overline{\text{RIO_TCLK}}$	RapidIO Transmit Clock	RapidIO	—	2	O	16-4/16-8
RIO_TD[0:7], $\overline{\text{RIO_TD}}$[0:7]	RapidIO Transmit Data	RapidIO	—	16	O	16-4/16-8
RIO_TFRAME, $\overline{\text{RIO_TFRAME}}$	RapidIO Transmit Frame	RapidIO	—	2	O	16-4/16-8
RIO_TX_CLK_IN $\overline{\text{RIO_TX_CLK_IN}}$	RapidIO Transmit Clock In	RapidIO	—	2	I	16-4/16-8
PCI_AD[63:0]	PCI Address/Data	PCI/X	—	64	I/O	15-1/15-7
PCI_C/$\overline{\text{BE}}$[7:0]	PCI Command/Byte Enable	PCI/X	—	8	I/O	15-1/15-7
PCI_PAR	PCI Parity	PCI/X	—	1	I/O	15-1/15-7

Table 3-1. MPC8560 Signal Reference by Functional Block (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
PCI_PAR64	PCI Parity 64	PCI/X	—	1	I/O	15-1/15-7
$\overline{\text{PCI_FRAME}}$	PCI Frame	PCI/X	—	1	I/O	15-1/15-7
$\overline{\text{PCI_TRDY}}$	PCI Target Ready	PCI/X	—	1	I/O	15-1/15-7
$\overline{\text{PCI_IRDY}}$	PCI Initiator Ready	PCI/X	—	1	I/O	15-1/15-7
$\overline{\text{PCI_STOP}}$	PCI Stop	PCI/X	—	1	I/O	15-1/15-7
$\overline{\text{PCI_DEVSEL}}$	PCI Device Select	PCI/X	—	1	I/O	15-1/15-7
PCI_IDSEL	PCI Initial Device Select	PCI/X	—	1	I	15-1/15-7
$\overline{\text{PCI_REQ64}}$	PCI Request 64	PCI/X	cfg_pci_width	1	I/O	15-1/15-7
$\overline{\text{PCI_ACK64}}$	PCI Acknowledge 64	PCI/X	—	1	I/O	15-1/15-7
$\overline{\text{PCI_PERR}}$	PCI Parity Error	PCI/X	—	1	I/O	15-1/15-7
$\overline{\text{PCI_SERR}}$	PCI System Error	PCI/X	—	1	I/O	15-1/15-7
$\overline{\text{PCI_REQ0}}$	PCI Request 0	PCI/X	—	1	I/O	15-1/15-7
$\overline{\text{PCI_REQ}}$[4:1]	PCI Request 4–1	PCI/X	—	4	I	15-1/15-7
$\overline{\text{PCI_GNT0}}$	PCI Grant 0	PCI/X	—	1	I/O	15-1/15-7
$\overline{\text{PCI_GNT}}$[4:1]	PCI Grant 4–1	PCI/X	cfg_pci_mode cfg_pci_debug cfg_pci_arbiter cfg_pci_impd	4	O	15-1/15-7
EC_GTX_CLK125	Gigabit Reference Clock	Gigabit Clock	—	1	I	13-1/13-10
EC_MDC	Ethernet Management Data Clock	Ethernet Management	cfg_tsec_reduce	1	O	13-1/13-10
EC_MDIO	Ethernet Management Data In/Out	Ethernet Management	—	1	I/O	13-1/13-10
TSEC1_TXD7	TSEC1 Transmit Data 7	TSEC1	deviceID0	1	O	13-1/13-10
TSEC1_TXD6	TSEC1 Transmit Data 6	TSEC1	deviceID1	1	O	13-1/13-10
TSEC1_TXD5	TSEC1 Transmit Data 5	TSEC1	deviceID2	1	O	13-1/13-10
TSEC1_TXD4	TSEC1 Transmit Data 4	TSEC1	deviceID3	1	O	13-1/13-10
TSEC1_TXD3	TSEC1 Transmit Data 3	TSEC1	—	1	O	13-1/13-10
TSEC1_TXD2	TSEC1 Transmit Data 2	TSEC1	—	1	O	13-1/13-10
TSEC1_TXD1	TSEC1 Transmit Data 1	TSEC1	—	1	O	13-1/13-10
TSEC1_TXD0	TSEC1 Transmit Data 0	TSEC1	—	1	O	13-1/13-10
TSEC1_TX_EN	TSEC1 Transmit Enable	TSEC1	—	1	O	13-1/13-10
TSEC1_TX_ER	TSEC1 Transmit Error	TSEC1	—	1	O	13-1/13-10
TSEC1_TX_CLK	TSEC1 Transmit Clock In	TSEC1	—	1	I	13-1/13-10
TSEC1_GTX_CLK	TSEC1 Transmit Clock Out	TSEC1	cfg_tsec1	1	O	13-1/13-10

Table 3-1. MPC8560 Signal Reference by Functional Block (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
TSEC1_CRS	TSEC1 Carrier Sense	TSEC1	—	1	I	13-1/13-10
TSEC1_COL	TSEC1 Collision Detect	TSEC1	—	1	I	13-1/13-10
TSEC1_RXD[7:0]	TSEC1 Receive Data	TSEC1	—	8	I	13-1/13-10
TSEC1_RX_DV	TSEC1 Receive Data Valid	TSEC1	—	1	I	13-1/13-10
TSEC1_RX_ER	TSEC1 Receiver Error	TSEC1	—	1	I	13-1/13-10
TSEC1_RX_CLK	TSEC1 Receive Clock	TSEC1	—	1	I	13-1/13-10
TSEC2_TXD7	TSEC2 Transmit Data 7	TSEC2	deviceID4	1	O	13-1/13-10
TSEC2_TXD6	TSEC2 Transmit Data 6	TSEC2	deviceID5	1	O	13-1/13-10
TSEC2_TXD5]	TSEC2 Transmit Data 5	TSEC2	deviceID6	1	O	13-1/13-10
TSEC2_TXD4	TSEC2 Transmit Data 4	TSEC2	deviceID7	1	O	13-1/13-10
TSEC2_TXD3]	TSEC2 Transmit Data 3	TSEC2	—	1	O	13-1/13-10
TSEC2_TXD2	TSEC2 Transmit Data 2	TSEC2	—	1	O	13-1/13-10
TSEC2_TXD1	TSEC2 Transmit Data 1	TSEC2	—	1	O	13-1/13-10
TSEC2_TXD0	TSEC2 Transmit Data 0	TSEC2	—	1	O	13-1/13-10
TSEC2_TX_EN	TSEC2 Transmit Enable	TSEC2	—	1	O	13-1/13-10
TSEC2_TX_ER	TSEC2 Transmit Error	TSEC2	—	1	O	13-1/13-10
TSEC2_TX_CLK	TSEC2 Transmit Clock In	TSEC2	—	1	I	13-1/13-10
TSEC2_GTX_CLK	TSEC2 Transmit Clock Out	TSEC2	cfg_tsec2	1	O	13-1/13-10
TSEC2_CRS	TSEC2 Carrier Sense	TSEC2	—	1	I	13-1/13-10
TSEC2_COL	TSEC2 Collision Detect	TSEC2	—	1	I	13-1/13-10
TSEC2_RXD[0:7]	TSEC2 Receive Data	TSEC2	—	8	I	13-1/13-10
TSEC2_RX_DV	TSEC2 Receive Data Valid	TSEC2	—	1	I	13-1/13-10
TSEC2_RX_ER	TSEC2 Receiver Error	TSEC2	—	1	I	13-1/13-10
TSEC2_RX_CLK	TSEC2 Receive Clock	TSEC2	—	1	I	13-1/13-10
LAD[0:31]	LBC Address/Data	LBC	—	32	I/O	12-1/12-5
LDP[0:3]	LBC Data Parity	LBC	—	4	I/O	12-1/12-5
LA27	LBC Burst Address	LBC	cfg_cpu_boot	1	O	12-1/12-5
LA[28:31]	LBC Port Address	LBC	cfg_sys_pll0 cfg_sys_pll1 cfg_sys_pll2 cfg_sys_pll3	4	O	12-1/12-5
$\overline{\text{LCS}}$[0:2]	LBC Chip Select 0–2	LBC	cfg_rom_loc[0:2]	3	O	12-1/12-5
$\overline{\text{LCS}}$[3:4]	LBC Chip Select 3–4	LBC	—	2	O	12-1/12-5
$\overline{\text{LCS5}}$	LBC Chip Select 5	LBC	$\overline{\text{DMA_DREQ2}}$	1	O	12-1/12-5

Table 3-1. MPC8560 Signal Reference by Functional Block (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
LCS6	LBC Chip Selec 6t	LBC	DMA_DACK2	1	O	12-1/12-5
LCS7	LBC Chip Select 7	LBC	DMA_DDONE2	1	O	12-1/12-5
LWE[0:1]/ LSDDQM[0:1]/ LBS[0:1]	LBC Write Enable/Byte Lane Data Mask/Byte Select 0–1	LBC	—	2	O	12-1/12-5
LWE[2:3]/ LSDDQM[2:3]/ LBS[2:3]	LBC Write Enable/Byte Lane Data Mask/Byte Select 2–3	LBC	cfg_host_agt0 cfg_host_agt1	2	O	12-1/12-5
LBCTL	LBC Data Buffer Control	LBC	—	1	O	12-1/12-5
LALE	LBC Address Latch Enable	LBC	cfg_core_pll0	1	O	12-1/12-5
LGPL0/LSDA10	LBC UPM General Purpose Line 0/SDRAM Address bit 10	LBC	cfg_rio_clk0	1	O	12-1/12-5
LGPL1/LSDWE	LBC GP Line 1/SDRAM Write Enable	LBC	cfg_rio_clk0	1	O	12-1/12-5
LGPL2/LOE/ LSDRAS	LBC GP Line 2/Output Enable/SDRAM RAS	LBC	cfg_core_pll1	1	O	12-1/12-5
LGPL3/LSDCAS	LBC GP Line 3/SDRAM CAS	LBC	cfg_boot_seq0	1	O	12-1/12-5
LGPL4/LGTA/ LUPWAIT/LPBSE	LBC GP Line 4/GPCM Terminate Access/UPM Wait/Parity Byte Select	LBC	—	1	I/O	12-1/12-5
LGPL5	LBC GP Line 5 Address	LBC	cfg_boot_seq1	1	O	12-1/12-5
LCKE	LBC Clock Enable	LBC	—	1	O	12-1/12-5
LCLK[0:2]	LBC Clock	LBC	—	3	O	12-1/12-5
LSYNC_IN, LSYNC_OUT	LBC DLL Synchronization	LBC	—	2	I/O	12-1/12-5
DMA_DREQ[0:1]	DMA Request 0–1	DMA	—	2	I	14-3/14-6
DMA_DREQ2	DMA Request 2	DMA	LCS5	1	I	14-3/14-6
DMA_DREQ3	DMA Request 3	DMA	IRQ9	1	I	14-3/14-6
DMA_DACK[0:1]	DMA Acknowledge 0–1	DMA	—	2	O	14-3/14-6
DMA_DACK2	DMA Acknowledge 2	DMA	LCS6	1	O	14-3/14-6
DMA_DACK3	DMA Acknowledge 3	DMA	IRQ10	1	O	14-3/14-6
DMA_DDONE[0:1]	DMA Done 0–1	DMA	—	2	O	14-3/14-6
DMA_DDONE2	DMA Done 2	DMA	LCS7	1	O	14-3/14-6
DMA_DDONE3	DMA Done 3	DMA	IRQ11	1	O	14-3/14-6
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

Table 3-1. MPC8560 Signal Reference by Functional Block (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
IRQ[9]	External Interrupt 9	PIC	$\overline{\text{DMA_DREQ3}}$	1	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
$\overline{\text{IRQ_OUT}}$	Interrupt Output	PIC	—	1	O	10-3/10-8
IIC_SDA	I^2C Serial Data	I^2C	—	1	I/O	11-2/11-4
IIC_SCL	I^2C Serial Clock	I^2C	—	1	I/O	11-2/11-4
$\overline{\text{HRESET}}$	Hard Reset	System Control	—	1	I	4-1/4-1
$\overline{\text{HRESET_REQ}}$	Hard Reset Request	System Control	—	1	O	4-1/4-1
$\overline{\text{SRESET}}$	Soft Reset	System Control	—	1	I	4-1/4-1
$\overline{\text{CKSTP_IN}}$	Checkstop In	System Control	—	1	I	17-2/17-3
$\overline{\text{CKSTP_OUT}}$	Checkstop Out	System Control	—	1	O	17-2/17-3
TRIG_IN	Watchpoint Trigger In	Debug	—	1	I	19-1/19-5
TRIG_OUT	Watchpoint Trigger Out	Debug	—	1	I/O	19-1/19-5
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	19-1/19-5
MDVAL	Memory Debug Data Valid	Debug	—	1	O	19-1/19-5
ASLEEP	Asleep	Power Mgmt	—	1	O	17-2/17-3
SYSCLK	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	17-2/17-3
$\overline{\text{LSSD_MODE}}$	LSSD Mode	Test	—	1	I	19-1/19-5
L1_TSTCLK	L1 Test Clock	Test	—	1	I	19-1/19-5
L2_TSTCLK	L2 Test Clock	Test	—	1	I	19-1/19-5
TEST_SEL	Test Select	Test	—	1	I	19-1/19-5
TCK	Test Clock	JTAG	—	1	I	19-1/19-5
TDI	Test Data In	JTAG	—	1	I	19-1/19-5
TDO	Test Data Out	JTAG	—	1	O	19-1/19-5
TMS	Test Mode Select	JTAG	—	1	I	19-1/19-5
$\overline{\text{TRST}}$	Test Reset	JTAG	—	1	I	19-1/19-5
PA[0:31]	Port A	CPM	See Chapter 35, “Parallel I/O Ports”	32	I/O	44-5/44-8

Table 3-1. MPC8560 Signal Reference by Functional Block (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
PB[4:31]	Port B	CPM	See Chapter 35, “Parallel I/O Ports”	28	I/O	44-6/44-12
PC[0:31]	Port C	CPM	See Chapter 35, “Parallel I/O Ports”	32	I/O	44-7/44-15
PD[4:31]	Port D	CPM	See Chapter 35, “Parallel I/O Ports”	28	I/O	44-8/44-17

Table 3-2. MPC8560 Alphabetical Signal Reference

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
ASLEEP	Asleep	Power Mgmt	—	1	O	17-2/17-3
$\overline{\text{CKSTP_IN}}$	Checkstop In	System Control	—	1	I	17-2/17-3
$\overline{\text{CKSTP_OUT}}$	Checkstop Out	System Control	—	1	O	17-2/17-3
CLK_OUT	Clock Out	Clock	—	1	O	17-2/17-3
$\overline{\text{DMA_DACK2}}$	DMA Acknowledge 2	DMA	$\overline{\text{LCS6}}$	1	O	14-3/14-6
$\overline{\text{DMA_DACK3}}$	DMA Acknowledge 3	DMA	IRQ10	1	O	14-3/14-6
$\overline{\text{DMA_DACK}}$[0:1]	DMA Acknowledge 0–1	DMA	—	2	O	14-3/14-6
$\overline{\text{DMA_DDONE2}}$	DMA Done 2	DMA	$\overline{\text{LCS7}}$	1	O	14-3/14-6
$\overline{\text{DMA_DDONE3}}$	DMA Done 3	DMA	IRQ11	1	O	14-3/14-6
$\overline{\text{DMA_DDONE}}$[0:1]	DMA Done 0–1	DMA	—	2	O	14-3/14-6
$\overline{\text{DMA_DREQ2}}$	DMA Request 2	DMA	$\overline{\text{LCS5}}$	1	I	14-3/14-6
$\overline{\text{DMA_DREQ3}}$	DMA Request 3	DMA	IRQ9	1	I	14-3/14-6
$\overline{\text{DMA_DREQ}}$[0:1]	DMA Request 0–1	DMA	—	2	I	14-3/14-6
EC_GTX_CLK125	Gigabit Reference Clock	Gigabit Clock	—	1	I	13-1/13-10
EC_MDC	Ethernet Management Data Clock	Ethernet Management	cfg_tsec_reduce	1	O	13-1/13-10
EC_MDIO	Ethernet Management Data In/Out	Ethernet Management	—	1	I/O	13-1/13-10
$\overline{\text{HRESET}}$	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^2C Serial Clock	I^2C	—	1	I/O	11-2/11-4
IIC_SDA	I^2C Serial Data	I^2C	—	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

Table 3-2. MPC8560 Alphabetical Signal Reference (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
IRQ[11]	External Interrupt 11	PIC	DMA_DDONE3	1	I	10-3/10-8
IRQ[9]	External Interrupt 9	PIC	DMA_DREQ3	1	I	10-3/10-8
IRQ_OUT	Interrupt Output	PIC	—	1	O	10-3/10-8
L1_TSTCLK	L1 Test Clock	Test	—	1	I	19-1/19-5
L2_TSTCLK	L2 Test Clock	Test	—	1	I	19-1/19-5
LA27	LBC Burst Address	LBC	cfg_cpu_boot	1	O	12-1/12-5
LAD[0:31]	LBC Address/Data	LBC	—	32	I/O	12-1/12-5
LALE	LBC Address Latch Enable	LBC	cfg_core_pll0	1	O	12-1/12-5
LA[28:31]	LBC Port Address	LBC	cfg_sys_pll0 cfg_sys_pll1 cfg_sys_pll2 cfg_sys_pll3	4	O	12-1/12-5
LBCTL	LBC Data Buffer Control	LBC	—	1	O	12-1/12-5
LCKE	LBC Clock Enable	LBC	—	1	O	12-1/12-5
LCLK[0:2]	LBC Clock	LBC	—	3	O	12-1/12-5
LCS5	LBC Chip Select 5	LBC	DMA_DREQ2	1	O	12-1/12-5
LCS6	LBC Chip Selec 6t	LBC	DMA_DACK2	1	O	12-1/12-5
LCS7	LBC Chip Select 7	LBC	DMA_DDONE2	1	O	12-1/12-5
LCS[0:2]	LBC Chip Select 0–2	LBC	cfg_rom_loc[0:2]	3	O	12-1/12-5
LCS[3:4]	LBC Chip Select 3–4	LBC	—	2	O	12-1/12-5
LDP[0:3]	LBC Data Parity	LBC	—	4	I/O	12-1/12-5
LGPL0/LSDA10	LBC UPM General Purpose Line 0/SDRAM Address bit 10	LBC	cfg_rio_clk0	1	O	12-1/12-5
LGPL1/LSDWE	LBC GP Line 1/SDRAM Write Enable	LBC	cfg_rio_clk0	1	O	12-1/12-5
LGPL2/LOE/ LSDRAS	LBC GP Line 2/Output Enable/SDRAM RAS	LBC	cfg_core_pll1	1	O	12-1/12-5
LGPL3/LSDCAS	LBC GP Line 3/SDRAM CAS	LBC	cfg_boot_seq0	1	O	12-1/12-5
LGPL4/LGTA/ LUPWAIT/LPBSE	LBC GP Line 4/GPCM Terminate Access/UPM Wait/Parity Byte Select	LBC	—	1	I/O	12-1/12-5
LGPL5	LBC GP Line 5 Address	LBC	cfg_boot_seq1	1	O	12-1/12-5
LSSD_MODE	LSSD Mode	Test	—	1	I	19-1/19-5
LSYNC_IN, LSYNC_OUT	LBC DLL Synchronization	LBC	—	2	I/O	12-1/12-5

Table 3-2. MPC8560 Alphabetical Signal Reference (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
$\overline{\text{LWE}}$[0:1]/ LSDDQM[0:1]/ $\overline{\text{LBS}}$[0:1]	LBC Write Enable/Byte Lane Data Mask/Byte Select 0–1	LBC	—	2	O	12-1/12-5
$\overline{\text{LWE}}$[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	12-1/12-5
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], $\overline{\text{MCK}}$[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
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	19-1/19-5
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	19-1/19-5
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
PA[0:31]	Port A	CPM	See Chapter 35, “Parallel I/O Ports”	32	I/O	44-5/44-8
PB[4:31]	Port B	CPM	See Chapter 35, “Parallel I/O Ports”	28	I/O	44-6/44-12
$\overline{\text{PCI_ACK64}}$	PCI Acknowledge 64	PCI/X	—	1	I/O	15-1/15-7
PCI_AD[63:0]	PCI Address/Data	PCI/X	—	64	I/O	15-1/15-7
PCI_C/$\overline{\text{BE}}$[7:0]	PCI Command/Byte Enable	PCI/X	—	8	I/O	15-1/15-7

Table 3-2. MPC8560 Alphabetical Signal Reference (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
PCI_DEVSEL	PCI Device Select	PCI/X	—	1	I/O	15-1/15-7
PCI_FRAME	PCI Frame	PCI/X	—	1	I/O	15-1/15-7
PCI_GNT0	PCI Grant 0	PCI/X	—	1	I/O	15-1/15-7
PCI_GNT[4:1]	PCI Grant 4–1	PCI/X	cfg_pci_mode cfg_pci_debug cfg_pci_arbiter cfg_pci_impd	4	O	15-1/15-7
PCI_IDSEL	PCI Initial Device Select	PCI/X	—	1	I	15-1/15-7
PCI_IRDY	PCI Initiator Ready	PCI/X	—	1	I/O	15-1/15-7
PCI_PAR	PCI Parity	PCI/X	—	1	I/O	15-1/15-7
PCI_PAR64	PCI Parity 64	PCI/X	—	1	I/O	15-1/15-7
PCI_PERR	PCI Parity Error	PCI/X	—	1	I/O	15-1/15-7
PCI_REQ0	PCI Request 0	PCI/X	—	1	I/O	15-1/15-7
PCI_REQ64	PCI Request 64	PCI/X	cfg_pci_width	1	I/O	15-1/15-7
PCI_REQ[4:1]	PCI Request 4–1	PCI/X	—	4	I	15-1/15-7
PCI_SERR	PCI System Error	PCI/X	—	1	I/O	15-1/15-7
PCI_STOP	PCI Stop	PCI/X	—	1	I/O	15-1/15-7
PCI_TRDY	PCI Target Ready	PCI/X	—	1	I/O	15-1/15-7
PC[0:31]	Port C	CPM	See Chapter 35, “Parallel I/O Ports”	32	I/O	44-7/44-15
PD[4:31]	Port D	CPM	See Chapter 35, “Parallel I/O Ports”	28	I/O	44-8/44-17
RIO_RCLK, RIO_RCLK	RapidIO Receive Clocks	RapidIO	—	2	I	16-4/16-8
RIO_RD[0:7], RIO_RD[0:7]	RapidIO Receive Data	RapidIO	—	16	I	16-4/16-8
RIO_RFRAME, RIO_RFRAME	RapidIO Receive Frame	RapidIO	—	2	I	16-4/16-8
RIO_TCLK, RIO_TCLK	RapidIO Transmit Clock	RapidIO	—	2	O	16-4/16-8
RIO_TD[0:7], RIO_TD[0:7]	RapidIO Transmit Data	RapidIO	—	16	O	16-4/16-8
RIO_TFRAME, RIO_TFRAME	RapidIO Transmit Frame	RapidIO	—	2	O	16-4/16-8
RIO_TX_CLK_IN RIO_TX_CLK_IN	RapidIO Transmit Clock In	RapidIO	—	2	I	16-4/16-8
RTC	Real Time Clock	Clock	—	1	I	4-2/4-2

Table 3-2. MPC8560 Alphabetical Signal Reference (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
$\overline{\text{SRESET}}$	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	19-1/19-5
TDI	Test Data In	JTAG	—	1	I	19-1/19-5
TDO	Test Data Out	JTAG	—	1	O	19-1/19-5
TEST_SEL	Test Select	Test	—	1	I	19-1/19-5
TMS	Test Mode Select	JTAG	—	1	I	19-1/19-5
TRIG_IN	Watchpoint Trigger In	Debug	—	1	I	19-1/19-5
TRIG_OUT	Watchpoint Trigger Out	Debug	—	1	I/O	19-1/19-5
$\overline{\text{TRST}}$	Test Reset	JTAG	—	1	I	19-1/19-5
TSEC1_COL	TSEC1 Collision Detect	TSEC1	—	1	I	13-1/13-10
TSEC1_CRS	TSEC1 Carrier Sense	TSEC1	—	1	I	13-1/13-10
TSEC1_GTX_CLK	TSEC1 Transmit Clock Out	TSEC1	cfg_tsec1	1	O	13-1/13-10
TSEC1_RXD[7:0]	TSEC1 Receive Data	TSEC1	—	8	I	13-1/13-10
TSEC1_RX_CLK	TSEC1 Receive Clock	TSEC1	—	1	I	13-1/13-10
TSEC1_RX_DV	TSEC1 Receive Data Valid	TSEC1	—	1	I	13-1/13-10
TSEC1_RX_ER	TSEC1 Receiver Error	TSEC1	—	1	I	13-1/13-10
TSEC1_TXD0	TSEC1 Transmit Data 0	TSEC1	—	1	O	13-1/13-10
TSEC1_TXD1	TSEC1 Transmit Data 1	TSEC1	—	1	O	13-1/13-10
TSEC1_TXD2	TSEC1 Transmit Data 2	TSEC1	—	1	O	13-1/13-10
TSEC1_TXD3	TSEC1 Transmit Data 3	TSEC1	—	1	O	13-1/13-10
TSEC1_TXD4	TSEC1 Transmit Data 4	TSEC1	deviceID3	1	O	13-1/13-10
TSEC1_TXD5	TSEC1 Transmit Data 5	TSEC1	deviceID2	1	O	13-1/13-10
TSEC1_TXD6	TSEC1 Transmit Data 6	TSEC1	deviceID1	1	O	13-1/13-10
TSEC1_TXD7	TSEC1 Transmit Data 7	TSEC1	deviceID0	1	O	13-1/13-10
TSEC1_TX_CLK	TSEC1 Transmit Clock In	TSEC1	—	1	I	13-1/13-10
TSEC1_TX_EN	TSEC1 Transmit Enable	TSEC1	—	1	O	13-1/13-10
TSEC1_TX_ER	TSEC1 Transmit Error	TSEC1	—	1	O	13-1/13-10
TSEC2_COL	TSEC2 Collision Detect	TSEC2	—	1	I	13-1/13-10
TSEC2_CRS	TSEC2 Carrier Sense	TSEC2	—	1	I	13-1/13-10
TSEC2_GTX_CLK	TSEC2 Transmit Clock Out	TSEC2	cfg_tsec2	1	O	13-1/13-10
TSEC2_RXD[0:7]	TSEC2 Receive Data	TSEC2	—	8	I	13-1/13-10
TSEC2_RX_CLK	TSEC2 Receive Clock	TSEC2	—	1	I	13-1/13-10

Table 3-2. MPC8560 Alphabetical Signal Reference (continued)

Signal	Signal Name	Functional Block	Alternate Function(s)	#	I/O	Table/ Page
TSEC2_RX_DV	TSEC2 Receive Data Valid	TSEC2	—	1	I	13-1/13-10
TSEC2_RX_ER	TSEC2 Receiver Error	TSEC2	—	1	I	13-1/13-10
TSEC2_TXD0	TSEC2 Transmit Data 0	TSEC2	—	1	O	13-1/13-10
TSEC2_TXD1	TSEC2 Transmit Data 1	TSEC2	—	1	O	13-1/13-10
TSEC2_TXD2	TSEC2 Transmit Data 2	TSEC2	—	1	O	13-1/13-10
TSEC2_TXD3]	TSEC2 Transmit Data 3	TSEC2	—	1	O	13-1/13-10
TSEC2_TXD4	TSEC2 Transmit Data 4	TSEC2	deviceID7	1	O	13-1/13-10
TSEC2_TXD5]	TSEC2 Transmit Data 5	TSEC2	deviceID6	1	O	13-1/13-10
TSEC2_TXD6	TSEC2 Transmit Data 6	TSEC2	deviceID5	1	O	13-1/13-10
TSEC2_TXD7	TSEC2 Transmit Data 7	TSEC2	deviceID4	1	O	13-1/13-10
TSEC2_TX_CLK	TSEC2 Transmit Clock In	TSEC2	—	1	I	13-1/13-10
TSEC2_TX_EN	TSEC2 Transmit Enable	TSEC2	—	1	O	13-1/13-10
TSEC2_TX_ER	TSEC2 Transmit Error	TSEC2	—	1	O	13-1/13-10
$\overline{\text{UDE}}$	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 *MPC8560 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 *MPC8560 Hardware Specification*.

Note that the multiplexing of various signals on the MPC8560 is controlled by the PMUXCR register described in Chapter 17, “Global Utilities.” Also, the multiplexing of the CPM signals occurs through the CPM programming model. See Chapter 42, “Parallel I/O Ports,” for details on CPM signal multiplexing.

Table 3-3. MPC8560 Reset Configuration Signals

Functional Interface	Functional Signal Name	Reset Configuration Name	Default	Table/Page
PCI	PCI_REQ64	cfg_pci_width	1	15-1/15-7
	PCI_GNT4	cfg_pci_mode	1	15-1/15-7
	PCI_GNT3	cfg_pci_debug	1	15-1/15-7
	PCI_GNT2	cfg_pci_arbiter	1	15-1/15-7
	PCI_GNT1	cfg_pci_impd	1	15-1/15-7
Ethernet Management	EC_MDC	cfg_tsec_reduce	1	13-1/13-10
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	
LBC	LA27	cfg_cpu_boot	1	12-1/12-5
	LA[28:31]	cfg_sys_pll[0:3]	must be driven	12-1/12-5
	LCS[0:2]	cfg_rom_loc[0:2]	111	12-1/12-5
	LWE[2:3]/LBS[2:3]	cfg_host_agt0 cfg_host_agt1	11	12-1/12-5
	LALE	cfg_core_pll0	must be driven	12-1/12-5
	LGPL0/LSDA10	cfg_rio_clk0	1	12-1/12-5
	LGPL1/LSDWE	cfg_rio_clk1	1	12-1/12-5
	LGPL2/LOE/LSDRAS	cfg_core_pll1	must be driven	12-1/12-5
	LGPL3/LSDCAS	cfg_boot_seq0	1	12-1/12-5
	LGPL5/LSDAMUX	cfg_boot_seq1	1	12-1/12-5
Debug	MSRCID0	cfg_mem_debug	1	19-1/19-5
	MSRCID1	cfg_ddr_debug	1	19-1/19-5

3.4 Output Signal States During Reset

When a system reset is recognized (HRESET is asserted), the MPC8560 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 MPC8560 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]		
DDR Memory	MCKE[0:1]		
DDR Memory	MCK[0:5], $\overline{\text{MCK}}$[0:5]		—
RapidIO	RIO_TCLK, RIO_TCLK		
RapidIO	RIO_TD[0:7], $\overline{\text{RIO_TD}}$[0:7]		
RapidIO	RIO_TFRAME, $\overline{\text{RIO_TFRAME}}$		
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		
LBC	LCLK[0:2]		
LBC	$\overline{\text{LCS}}$[3:4]		
LBC	$\overline{\text{LWE}}$[0:1]/$\overline{\text{LBS}}$[0:1]		

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

Interface	Signal	State During Reset	Alt. Signal
LBC	LBCTL		
LBC	LALE		
LBC	LGPL0/LSDA10		
LBC	LGPL1/$\overline{\text{LSDWE}}$		
LBC	LGPL2/$\overline{\text{LOE}}$/$\overline{\text{LSDRAS}}$		
LBC	LGPL3/$\overline{\text{LSDCAS}}$		
DMA	$\overline{\text{DMA_DACK}}$[0:1]		—
DMA	$\overline{\text{DMA_DACK2}}$		$\overline{\text{LCS6}}$
DMA	$\overline{\text{DMA_DACK3}}$		IRQ10
DMA	$\overline{\text{DMA_DDONE}}$[0:1]		—
DMA	$\overline{\text{DMA_DDONE2}}$		$\overline{\text{LCS7}}$
DMA	$\overline{\text{DMA_DDONE3}}$		IRQ11
PIC	$\overline{\text{IRQ_OUT}}$		
System Control	$\overline{\text{HRESET_REQ}}$		
System Control	$\overline{\text{CKSTP_OUT}}$		
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 MPC8560.

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 MPC8560. Note that the $\overline{\text{CKSTP_IN}}$ and $\overline{\text{CKSTP_OUT}}$ signals are described in Chapter 17, “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 MPC8560 ($\overline{\text{HRESET}}$) causes the MPC8560 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 MPC8560 signals during reset.
		Timing	Assertion/Negation—See the *MPC8560 Hardware Specification* 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 MPC8560 (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 and a soft reset to the CPM. 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 and a soft reset to the CPM (level sensitive). 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 MPC8560. Note that there is also a CLK_OUT signal in the MPC8560; the signal driven on the CLK_OUT pin is selectable and described in Section 17.3.1.16, “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 MPC8560. 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 and the PLL in the CPMas 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 MPC8560 is in agent mode and when it is the host mode; the MPC8560 does not provide a separate PCI_CLK output in host mode.	
		Timing	Assertion/Negation—See the *MPC8560 Hardware Specification* for specific timing information for this signal.
RTC	I	Real time clock. The real time clock (RTC) signal is an optional input on the MPC8560. 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 *MPC8560 Hardware Specification* 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 MPC8560.

4.4 Functional Description

This section describes the various ways to reset the MPC8560 device, the power-on reset configurations, and the clocking on the MPC8560.

4.4.1 Reset Operations

There are two reset input signals on the MPC8560 that cause a hard and soft reset operation, respectively.

4.4.1.1 Hard Reset

The MPC8560 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 *MPC8560 Hardware Specification* for the timing requirements for $\overline{\text{HRESET}}$ assertion and negation.

The MPC8560 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 the $\overline{\text{SRESET}}$ signal causes the CPM to go through its soft reset sequence. In addition, 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 17.3.1.13, "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 MPC8560 is as follows:

1. Power is applied to meet the specifications in the *MPC8560 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 CPM reset signals (cpm_por_reset, cpm_hreset, and cpm_sreset) are asserted, and the e500 and CPM PLL configuartion inputs are applied, allowing the e500 and CPM PLLs to begin locking to the device clock (the CCB clock).
8. The CCB clock is cycled for approximately 50µs to lock the e500 PLL and the CPM 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 MPC8560 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.

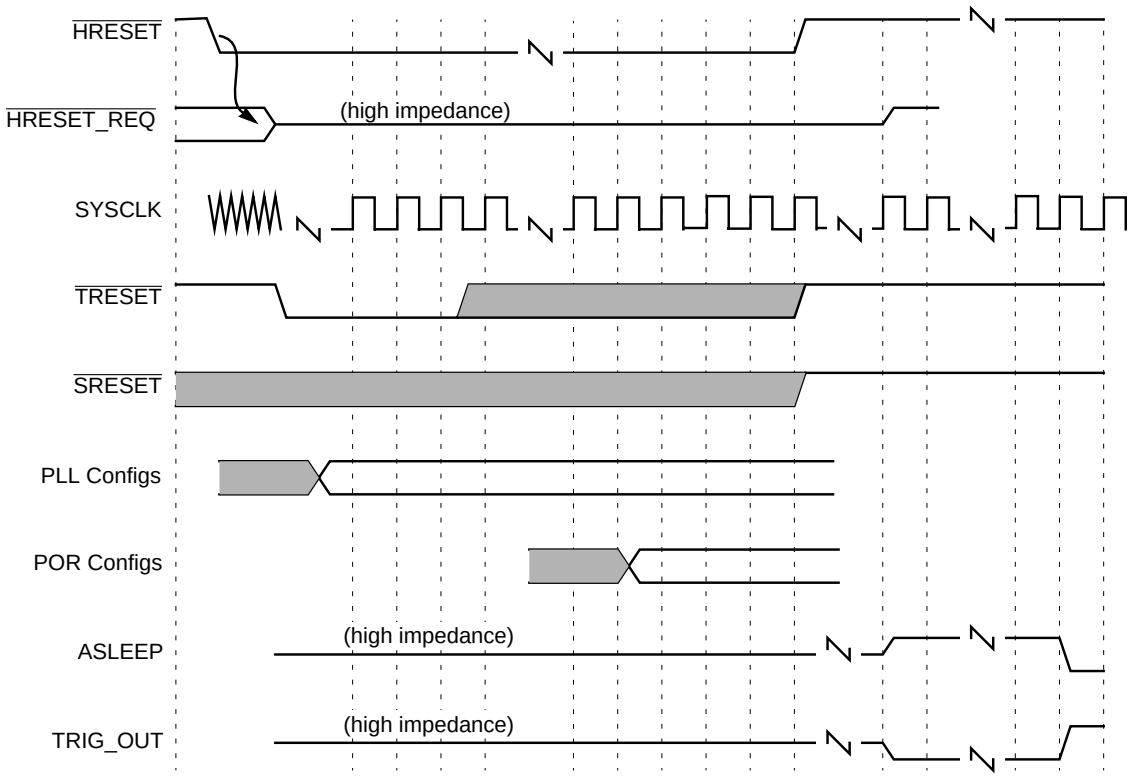


Figure 4-1. Power-on Reset Sequence

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 *MPC8560 Hardware Specificatio*n 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 MPC8560. 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 MPC8560 to act as a host or as an agent of a master on another interface. In host mode, the MPC8560 is immediately enabled to master transactions to the RapidIO and PCI interfaces. If the MPC8560 is an agent on the RapidIO or the PCI interfaces, then the MPC8560 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 MPC8560's interfaces appropriately. See details in the PCI and RapidIO programming models described in Chapter 15, "PCI Bus Interface," and Chapter 16, "RapidIO Interface," respectively.

Table 4-6. Host/Agent Configuration

Functional Signals	Reset Configuration Name	Value (binary)	Meaning
$\overline{\text{LWE}}$[2:3] Default (11)	cfg_host_agt[0:1]	00	MPC8560 acts as an agent of both a PCI/PCI-X and a RapidIO device.
		01	MPC8560 acts as an agent of a RapidIO host.
		10	MPC8560 acts as an agent of a PCI/PCI-X host.
		11	MPC8560 acts as the host processor (default).

NOTE

If the MPC8560 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 MPC8560 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 I^2C port before the host tries to configure the MPC8560. These options also specify normal or extended I^2C 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 Default (11)	cfg_boot_seq[0:1]	00	Reserved
		01	Normal I^2C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I^2C interface. A valid ROM must be present.
		10	Extended I^2C addressing mode is used. Boot sequencer is enabled and loads configuration information from a ROM on the I^2C interface. A valid ROM must be present.
		11	Boot sequencer is disabled. No I^2C 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 Default (1)	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.
		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 MPC8560 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
$\overline{\text{PCI_REQ64}}$	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 Impedence

Functional Signal	Reset Configuration Name	Value (binary)	Meaning
$\overline{\text{PCI_GNT1}}$	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
$\overline{\text{PCI_GNT2}}$	cfg_pci_arbiter	0	The on-chip PCI/PCI-X arbiter is disabled. External arbitration is required.
Default (1)		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
$\overline{\text{PCI_GNT3}}$ 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
$\overline{\text{PCI_GNT4}}$ 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 MPC8560 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 instrinsic delay of the output driver. Refer to the *MPC8560 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
$\overline{\text{LWE}}$[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] Default (11)	cfg_pci_hold[0:1]	11	0 added buffer delays--meets 0.7ns hold time requirement
		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 instrinsic delay of the output driver. Refer to the *MPC8560 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] Default (11)	cfg_lbiu_hold[0:1]	11	1 added buffer delay (0 added buffer delays for LALE)
		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 MPC8560 device.

4.4.4.1 System Clock/PCI Clock

The MPC8560 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 and the PLL in the CPM 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 MPC8560 is in agent mode and when it is the host mode; the MPC8560 does not provide a separate PCI_CLK output in host mode.

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

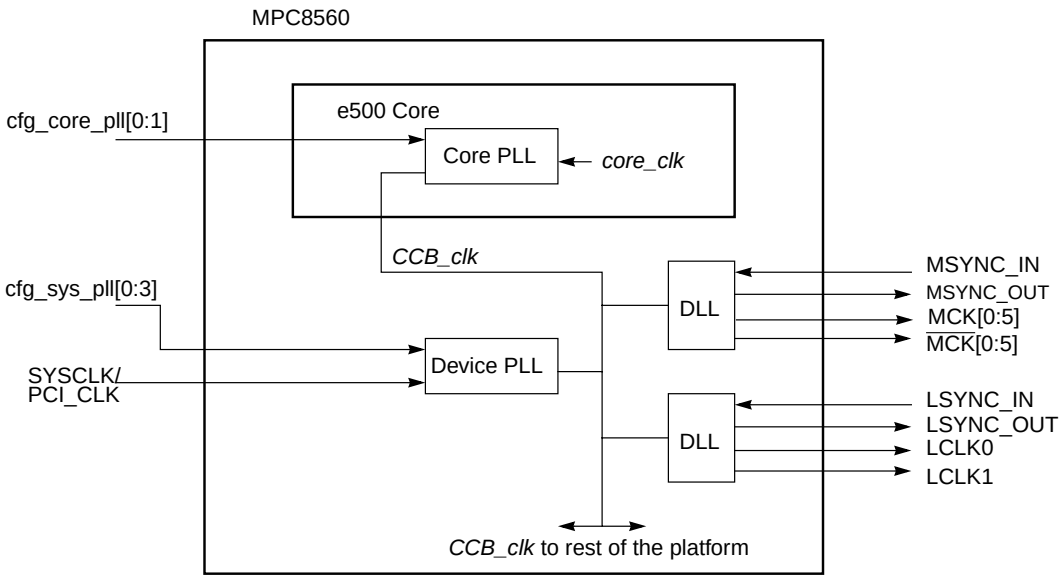


Figure 4-2. Clock Subsystem Block Diagram

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 MPC8560 RapidIO port is connected, then the transmit clock can be derived from the receive clock (RIO_RCLK and $\overline{\text{RIO_RCLK}}$).

If the MPC8560 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.

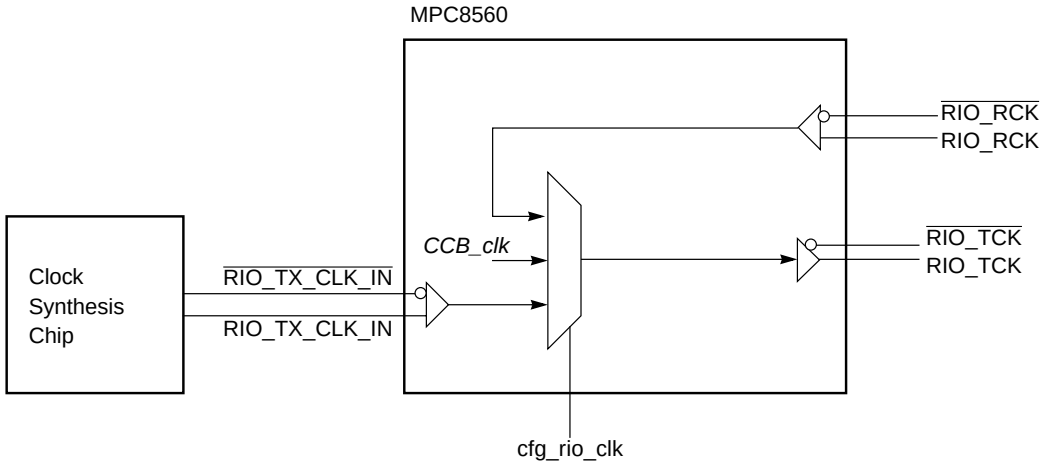


Figure 4-3. RapidIO Transmit Clock Options

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 MPC8560 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)

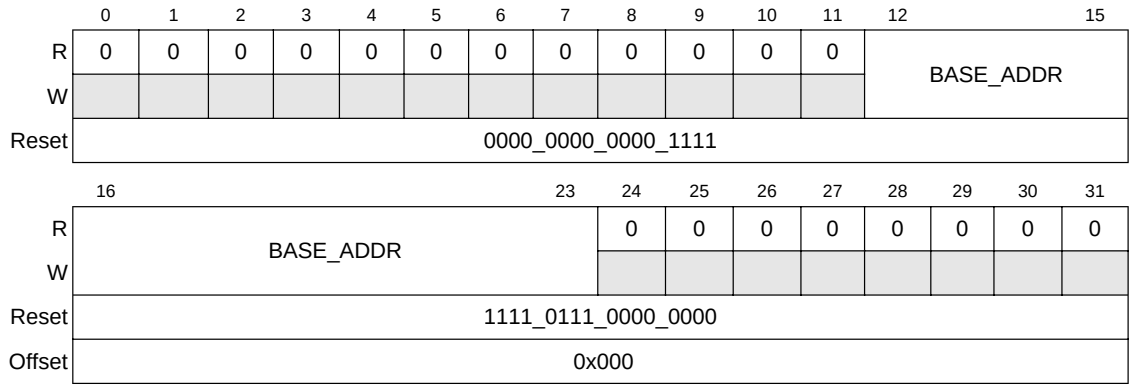


Figure 4-4. Configuration Control and Status Register Base Address Register (CCSRBAR)

Table 4-25. CCSRBAR Bit Settings

Bits	Name	Description
0–11	—	Write reserved, read = 0.
12–23	BASE_ADDR	Identifies the12 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 ALTCAR registers. These are intended to be used with the boot sequencer to allow the boot sequencer to access an alternate 1-Mbyte region of configuration space. By loading the proper boot sequencer command in the serial ROM the base address in the ALTCBAR can be combined with the 20 bits of address offset supplied from the serial ROM to generate a 32-bit address that is mapped to the target specified in ALTCAR. Thus, by configuring these registers, the boot sequencer has access to the entire memory map, one 1-Mbyte block at a time. See Section 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)

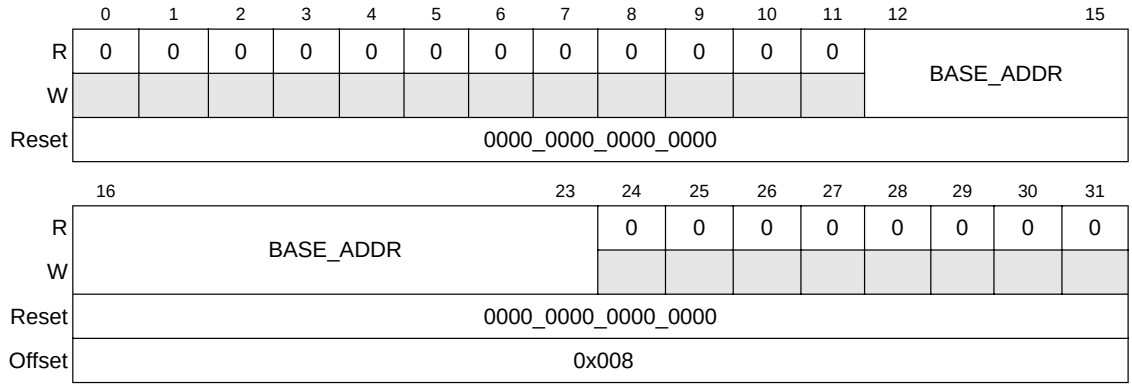


Figure 4-5. Alternate Configuration Base Address Register (ALTCBAR)

Table 4-26. ALTCBAR Bit Settings

Bits	Name	Description
0–11	—	Write reserved, read = 0
12–23	BASE_ADDR	Identifies the12 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)

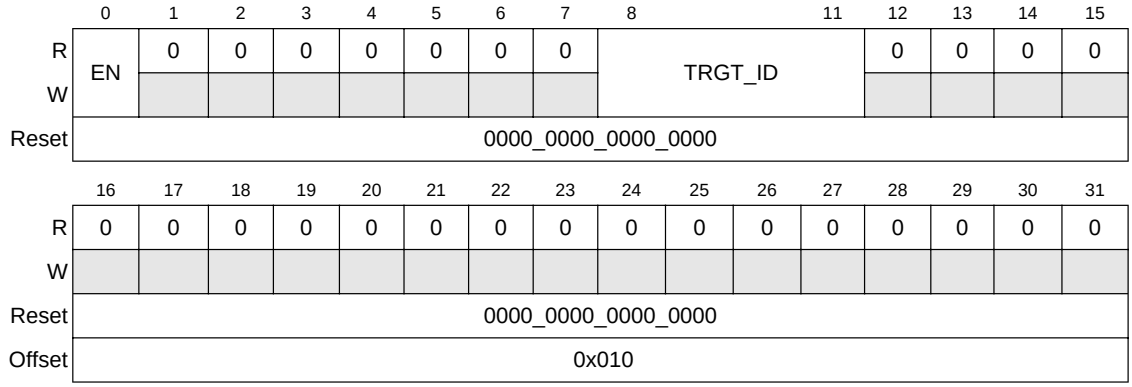


Figure 4-6. Alternate Configuration Attribute Register (ALTCCR)

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 MPC8560 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 MPC8560 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 MPC8560, 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)

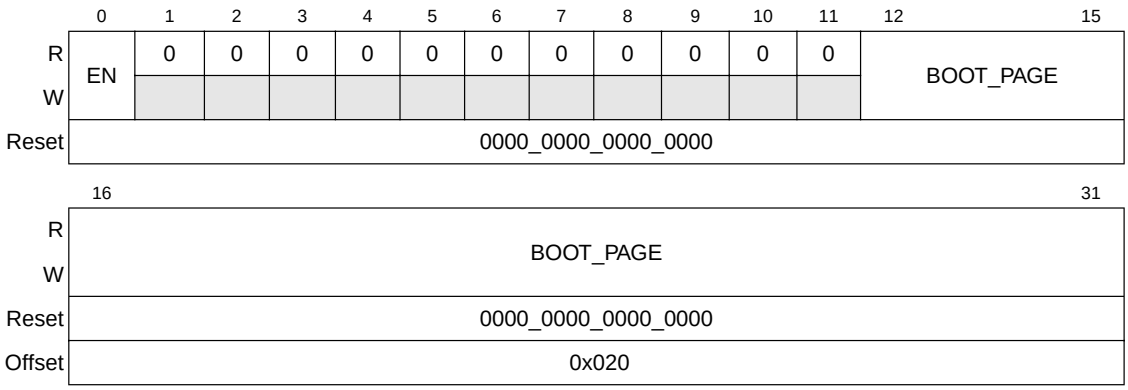


Figure 4-7. Boot Page Translation Register (BPTR)

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 I^2C 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

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 I^2C chapter.

Part II
e500 Core Complex and L2 Cache

This part describes the many features of the MPC8560 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 MPC8560. 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

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.

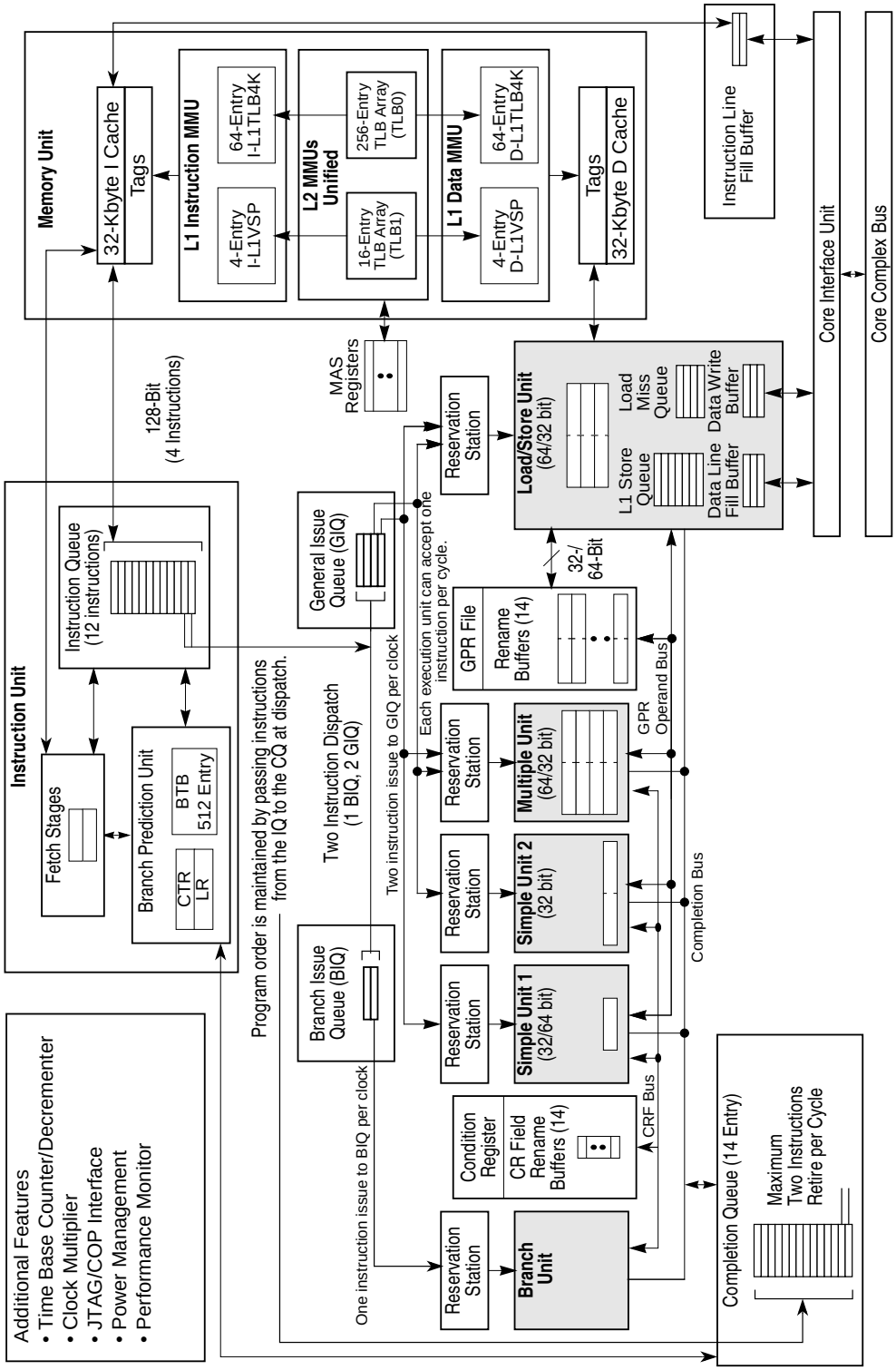


Figure 5-1. e500 Core Complex Block Diagram

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

– 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 **r**A or **r**B 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.

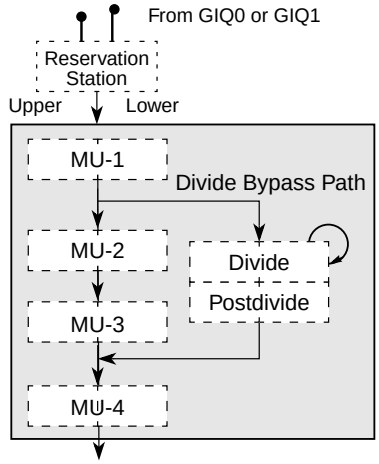


Figure 5-2. Four-Stage MU Pipeline, Showing the Divide Bypass

- 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 **r**A or **r**B 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.

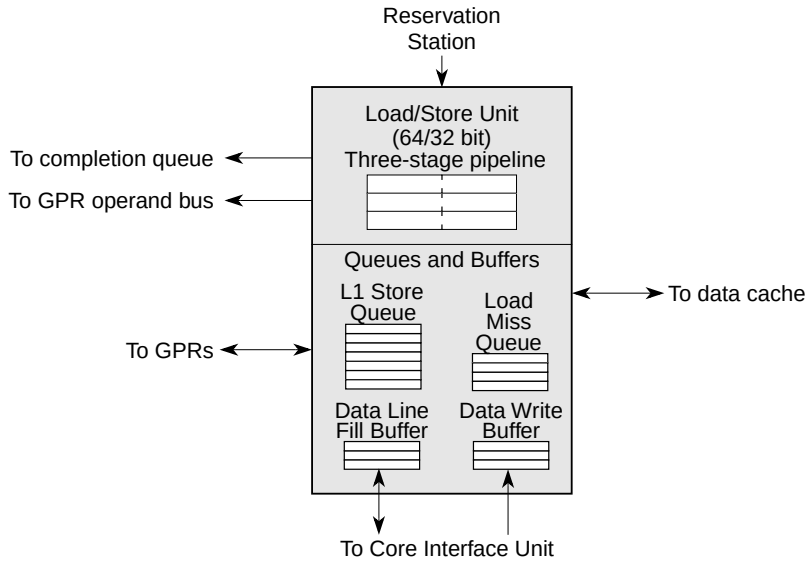


Figure 5-3. Three Stage Load/Store Unit

- 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

  - 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 MPC8560)

- 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**	**r**D,**r**A
Floating-Point Add	**efsadd**	**r**D,**r**A,**r**B
Convert Floating-Point from Signed Fraction	**efscfsf**	**r**D,**r**B
Convert Floating-Point from Signed Integer	**efscfsi**	**r**D,**r**B
Convert Floating-Point from Unsigned Fraction	**efscfuf**	**r**D,**r**B
Convert Floating-Point from Unsigned Integer	**efscfui**	**r**D,**r**B
Floating-Point Compare Equal	**efscmpeq**	**crf**D,**r**A,**r**B
Floating-Point Compare Greater Than	**efscmpgt**	**crf**D,**r**A,**r**B
Floating-Point Compare Less Than	**efscmplt**	**crf**D,**r**A,**r**B
Convert Floating-Point to Signed Fraction	**efsctsf**	**r**D,**r**B
Convert Floating-Point to Signed Integer	**efsctsi**	**r**D,**r**B
Convert Floating-Point to Signed Integer with Round toward Zero	**efsctsiz**	**r**D,**r**B
Convert Floating-Point to Unsigned Fraction	**efsctuf**	**r**D,**r**B
Convert Floating-Point to Unsigned Integer	**efsctui**	**r**D,**r**B
Convert Floating-Point to Unsigned Integer with Round toward Zero	**efsctui**	**r**D,**r**B
Floating-Point Divide	**efsdiv**	**r**D,**r**A,**r**B
Floating-Point Multiply	**efsmul**	**r**D,**r**A,**r**B
Floating-Point Negative Absolute Value	**efsnabs**	**r**D,**r**A
Floating-Point Negate	**efsneg**	**r**D,**r**A
Floating-Point Subtract	**efssub**	**r**D,**r**A,**r**B
Floating-Point Test Equal	**efststeq**	**crf**D,**r**A,**r**B
Floating-Point Test Greater Than	**efststgt**	**crf**D,**r**A,**r**B
Floating-Point Test Less Than	**efststlt**	**crf**D,**r**A,**r**B

— 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**	**r**D,PMRN
Move to Performance Monitor Register	**mtpmr**	PMRN,**r**S

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

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.

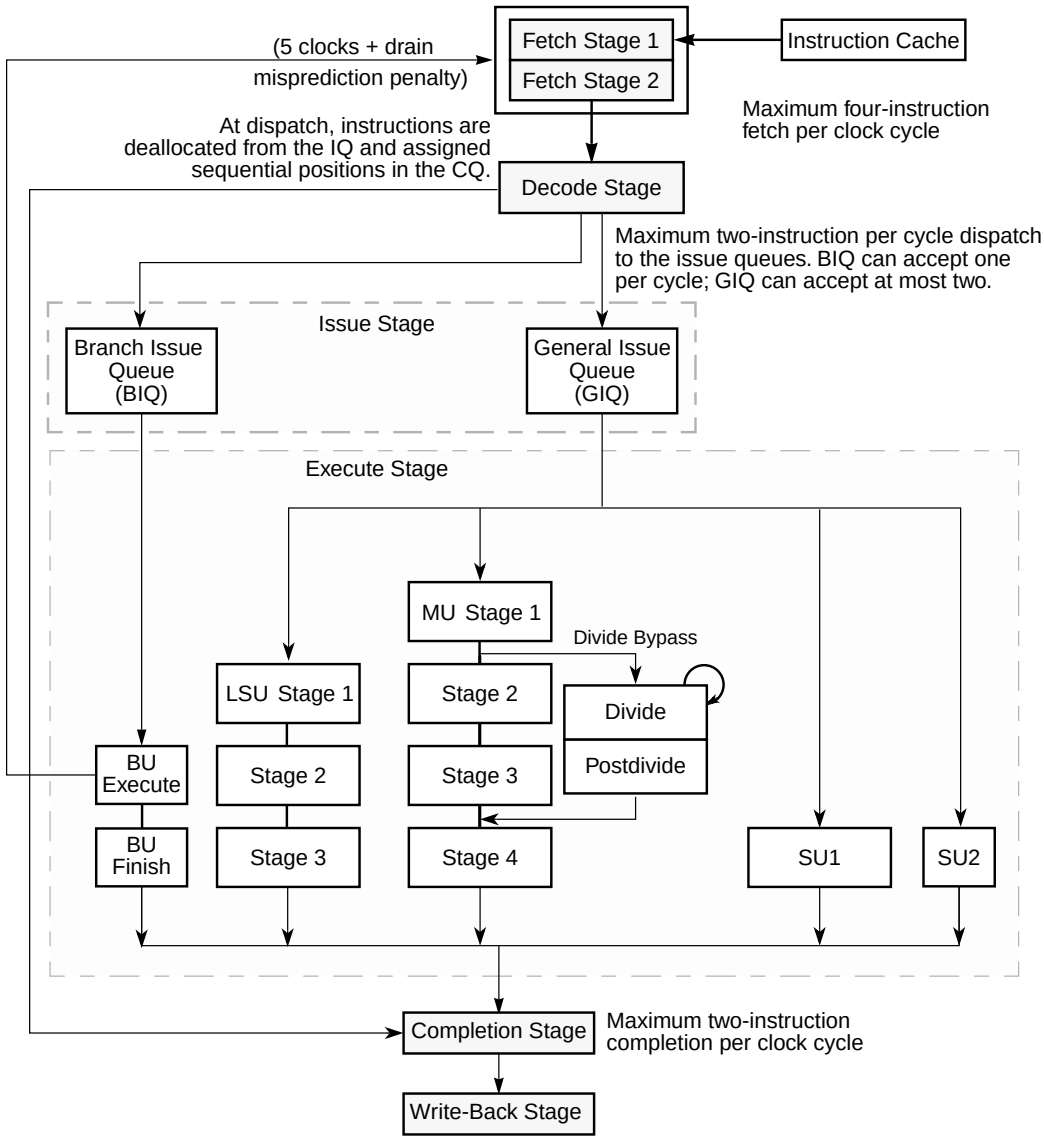


Figure 5-4. Instruction Pipeline Flow

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**, **nop**s, and some other instructions do not go to issue queues).
- The two issue queues, BIQ and GIQ, can accept as many as one and two instructions, respectively, in a cycle. 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.

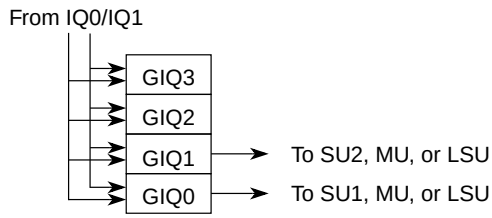


Figure 5-5. GPR Issue Queue (GIQ)

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.

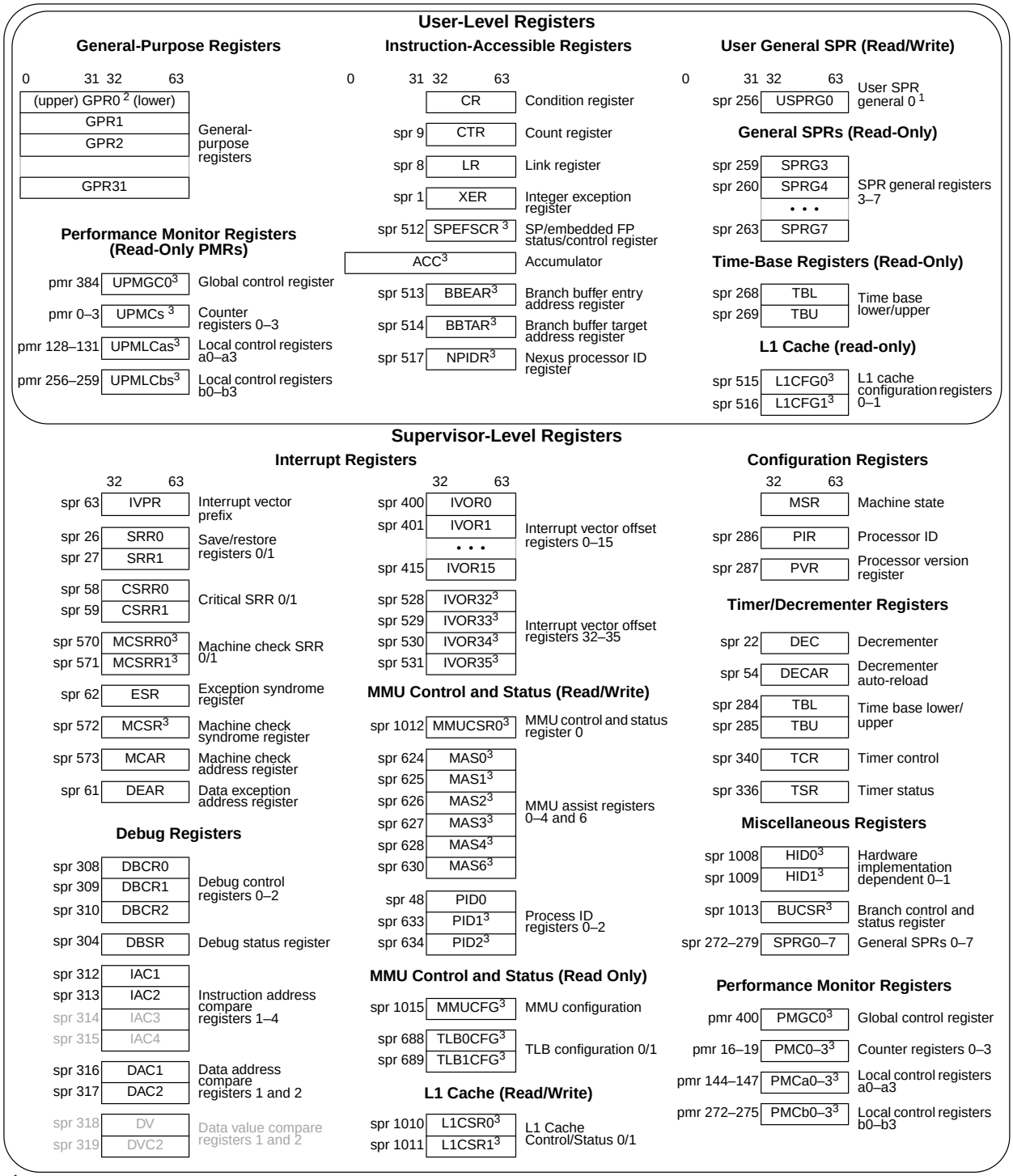
  The write-back stage occurs in the clock cycle after the instruction is retired.

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.

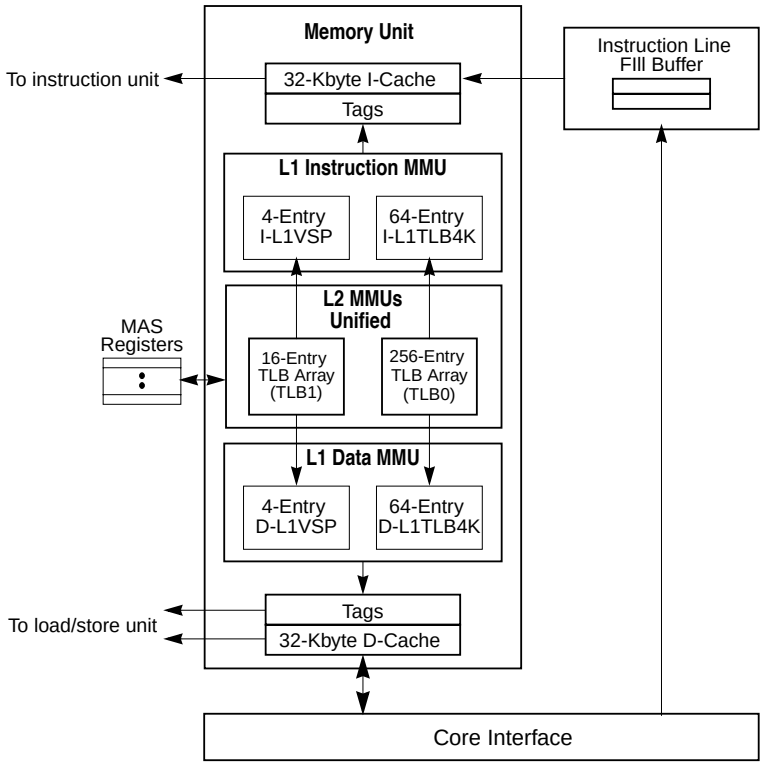


Figure 5-7. MMU Structure

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.

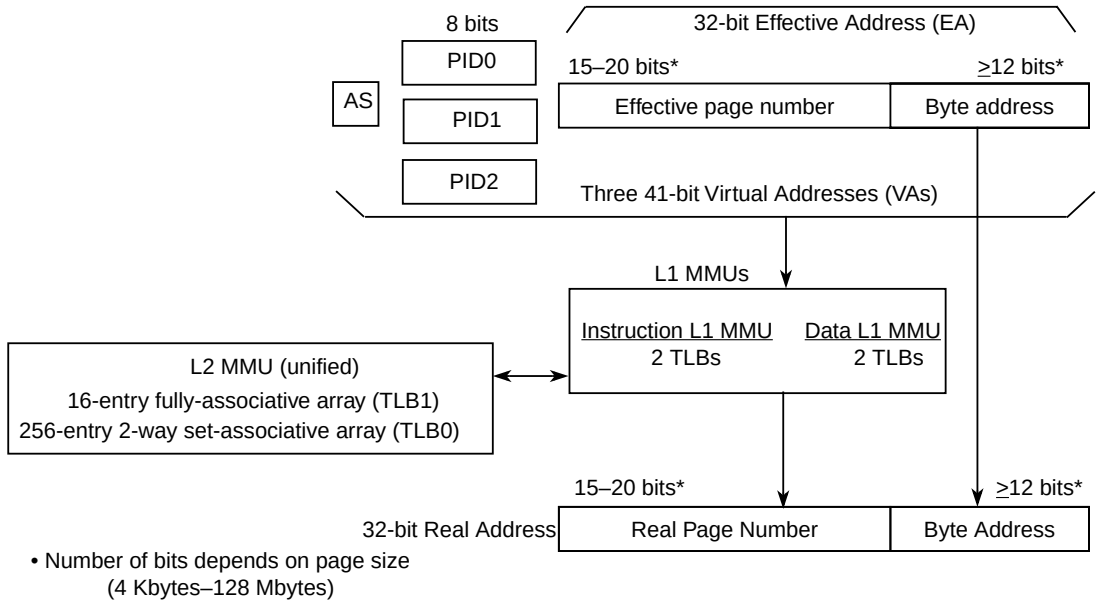


Figure 5-8. Effective-to-Real Address Translation Flow

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 decrementer 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.

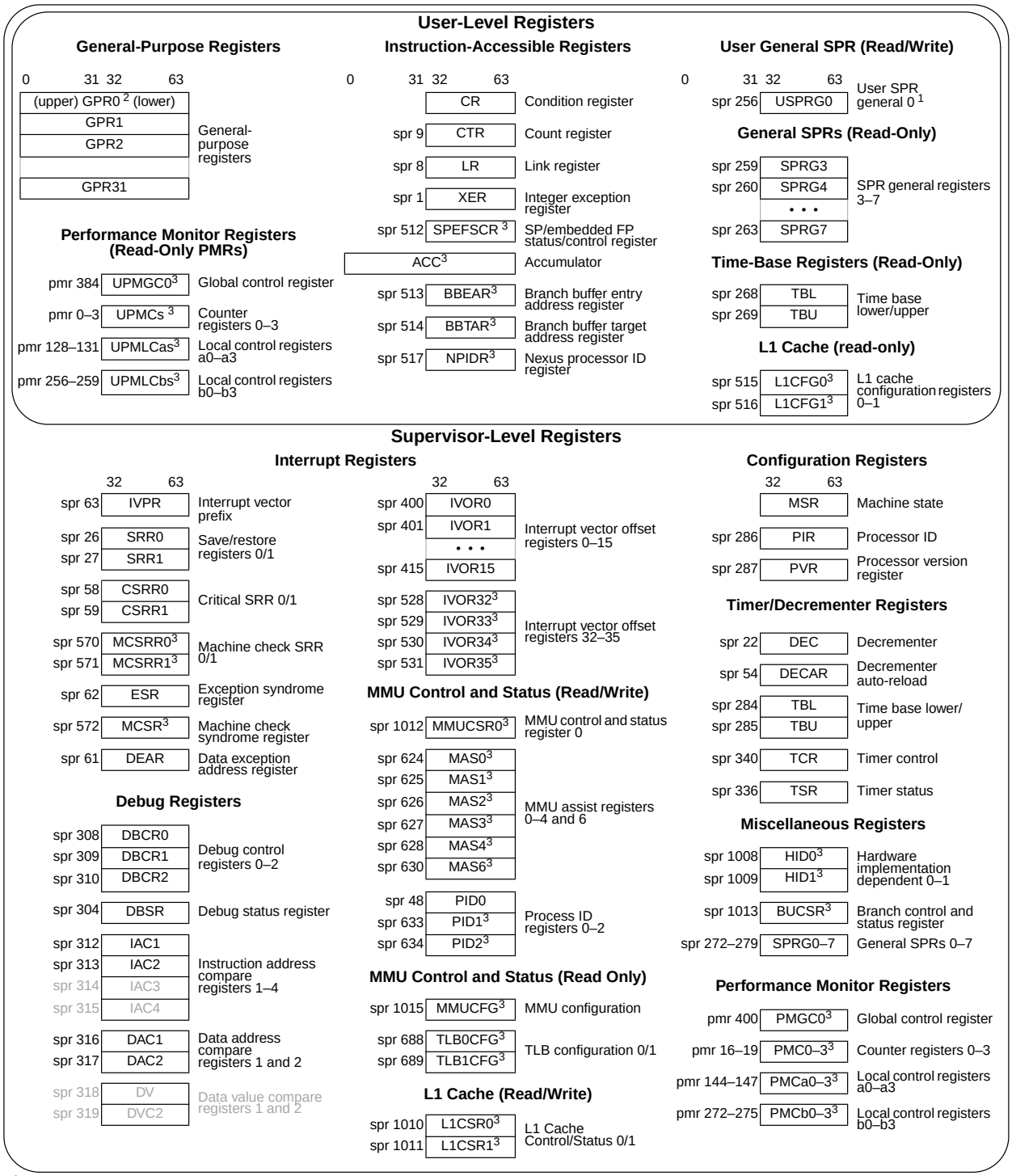
Chapter 6
e500 Register Summary

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 **mfspr** 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 **mtspr** and **mfspr** 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 Abbrevation)

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)

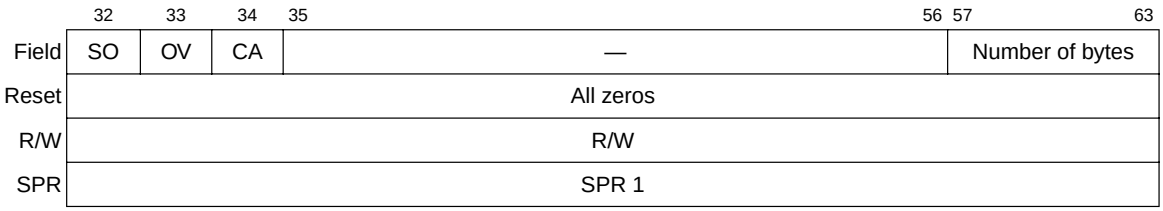


Figure 6-2. Integer Exception Register (XER)

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)

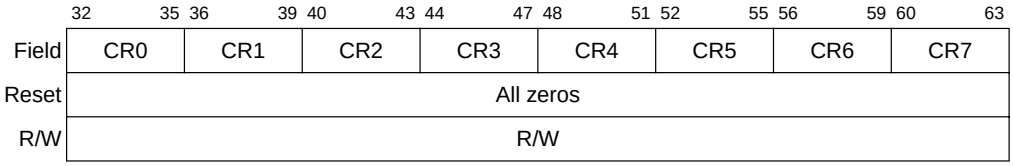


Figure 6-3. Condition Register (CR)

Table 6-4. BI Operand Settings for CR Fields

CR*n* Bits	CR Bits	BI	Description
CR0[0]	32	00000	Negative (LT)—Set when the result is negative. For SPE APU vector compare and vector test instructions: Set if the high-order element of **r**A is equal to the high-order element of **r**B; cleared otherwise.
CR0[1]	33	00001	Positive (GT)—Set when the result is positive (and not zero). For SPE APU vector compare and vector test instructions: Set if the low-order element of **r**A is equal to the low-order element of **r**B; cleared otherwise.
CR0[2]	34	00010	Zero (EQ)—Set when the result is zero. For SPE 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 **r**A is equal to the high-order element of **r**B; 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 **r**A is equal to the low-order element of **r**B; 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.
CR*n*[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: **r**A < SIMM or **r**B (signed comparison) or **r**A < UIMM or **r**B (unsigned comparison). For floating-point compare instructions: **fr**A < **fr**B. For SPE APU vector compare and vector test instructions: Set if the high-order element of **r**A is equal to the high-order element of **r**B; cleared otherwise.
CR*n*[1]	41 45 49 53 57 61	01001 01101 10001 10101 11001 11101	Greater than or floating-point greater than (GT, FG). For integer compare instructions: **r**A > SIMM or **r**B (signed comparison) or **r**A > UIMM or **r**B (unsigned comparison). For floating-point compare instructions: **fr**A > **fr**B. For SPE APU vector compare and vector test instructions: Set if the low-order element of **r**A is equal to the low-order element of **r**B; cleared otherwise.

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

CR*n* Bits	CR Bits	BI	Description
CR*n*[2]	42 46 50 54 58 62	01010 01110 10010 10110 11010 11110	Equal or floating-point equal (EQ, FE). For integer compare instructions: **r**A = SIMM, UIMM, or **r**B. For floating-point compare instructions: **fr**A = **fr**B. 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.
CR*n*[3]	43 47 51 55 59 63	01011 01111 10011 10111 11011 11111	Summary overflow or floating-point unordered (SO, FU). For integer compare instructions, this is a copy of XER[SO] at the completion of the instruction. For floating-point compare instructions, one or both of **fr**A and **fr**B is a NaN. 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.

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, decrementer, and fixed-interval timer interrupts are disabled. 1 External input, decrementer, 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)

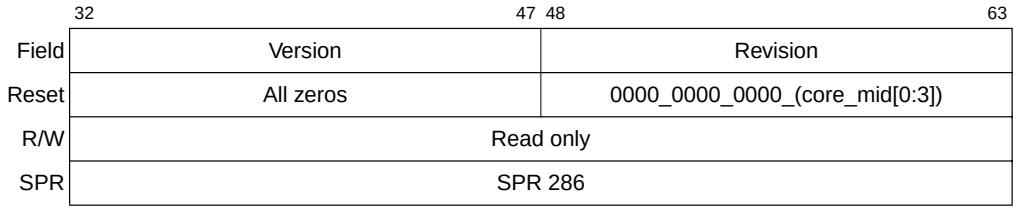


Figure 6-7. Processor ID Register (PIR)

6.5.3 Processor Version Register (PVR)

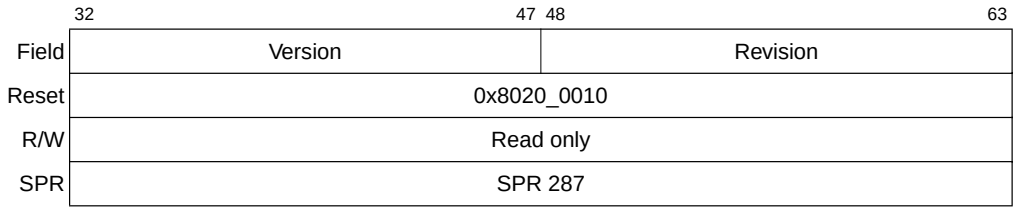


Figure 6-8. Processor Version Register (PVR)

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)

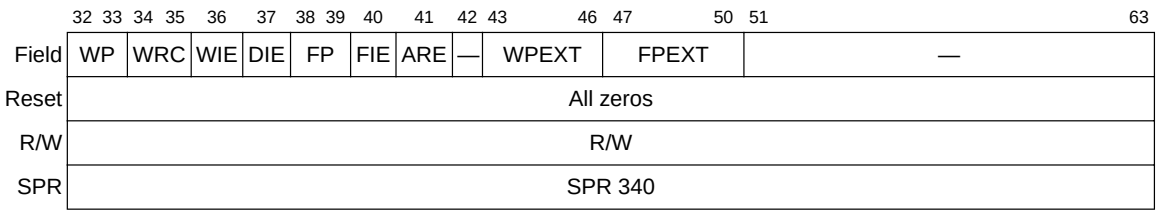


Figure 6-9. Timer Control Register (TCR)

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 **mfspr** and cleared by using **mtspr** 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

	32	63
Field	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)

	32	63
Field	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)

	32	63
Field	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)

	32	63
Field	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)

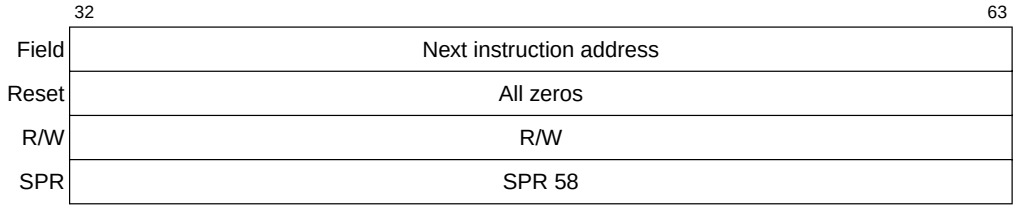


Figure 6-16. Critical Save/Restore Register 0 (CSRR0)

6.7.1.4 Critical Save/Restore Register 1 (CSRR1)

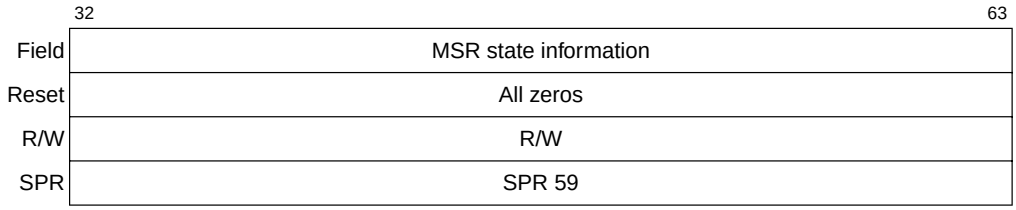


Figure 6-17. Critical Save/Restore Register 1 (CSRR1)

6.7.1.5 Data Exception Address Register (DEAR)

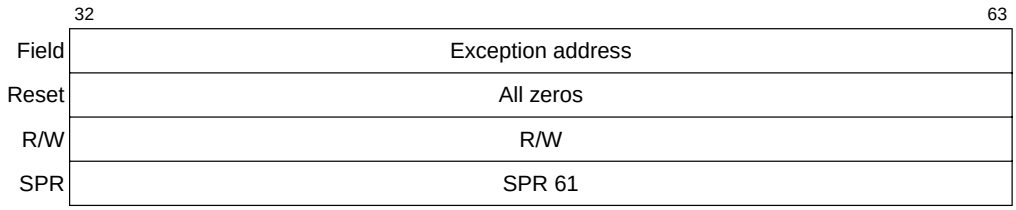


Figure 6-18. Data Exception Address Register (DEAR)

6.7.1.6 Interrupt Vector Prefix Register (IVPR)

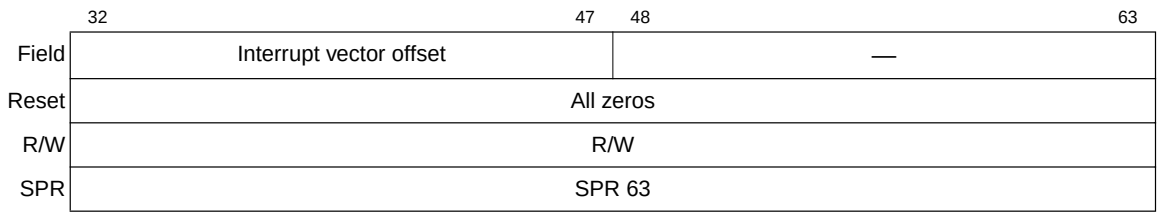


Figure 6-19. Interrupt Vector Prefix Register (IVPR)

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 e500does 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_BTB	—
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_BTB	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 R/W	SPRG0	272	Read/Write	Supervisor	
	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)

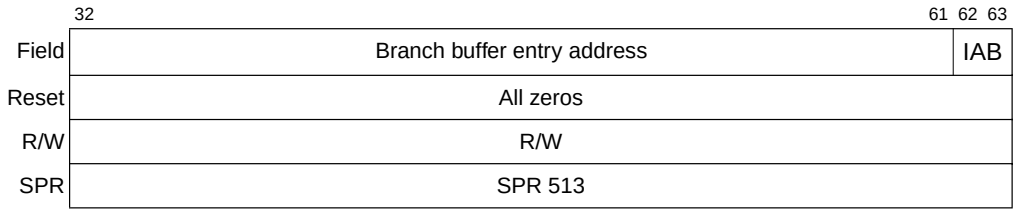


Figure 6-26. Branch Buffer Entry Address Register (BBEAR)

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)

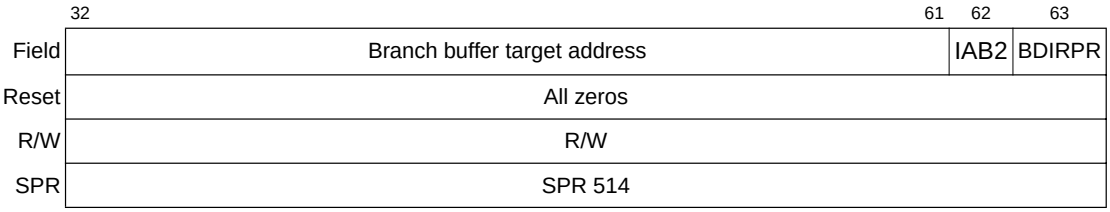


Figure 6-27. Branch Buffer Target Address Register (BBTAR)

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)

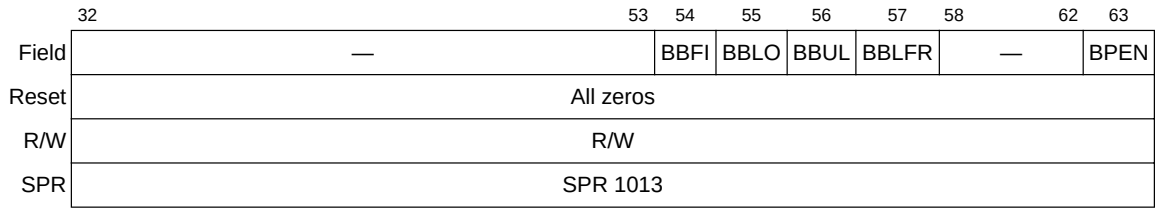


Figure 6-28. Branch Unit Control and Status Register (BUCSR)

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)

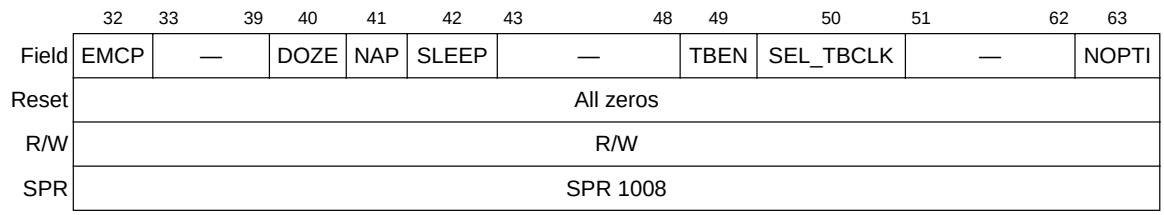


Figure 6-29. HID0 Register

Table 6-15. HID0 Field Descriptions

Bits	Name	Description
32	EMCP	Enable machine check pin, $\overline{\text{MCP}}$. Used to mask out further machine check exceptions caused by assertion of $\overline{\text{MCP}}$. 0 $\overline{\text{MCP}}$ is disabled. 1 $\overline{\text{MCP}}$ is enabled. If MSE[ME] = 0, asserting $\overline{\text{MCP}}$ causes checkstop. If MSR[ME] = 1, asserting $\overline{\text{MCP}}$ causes a machine check exception.
33–39	—	Reserved, should be cleared.
40	DOZE	Doze power management mode. If MSR[WE] is set, this bit controls DOZE mode. 0 Core not in doze mode 1 Core in doze mode
41	NAP	Nap power management mode. If MSR[WE] is set, this bit controls NAP mode. 0 Core not in nap mode 1 Core in nap mode
42	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)

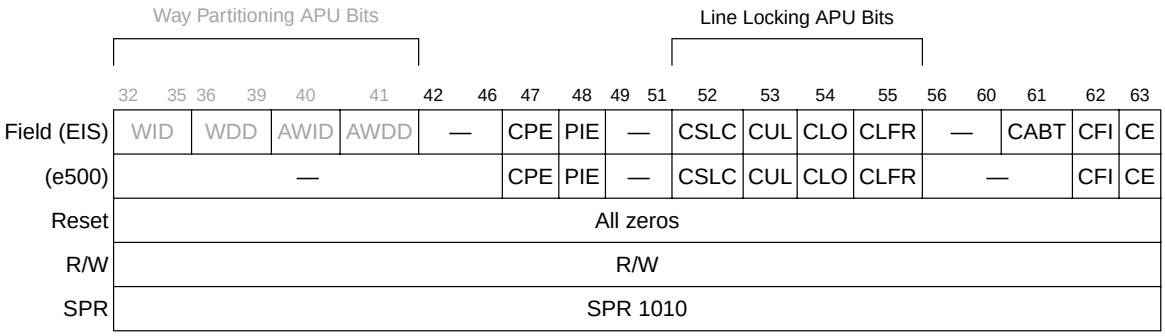


Figure 6-31. L1 Cache Control and Status Register 0 (L1CSR0)

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 aflash 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)

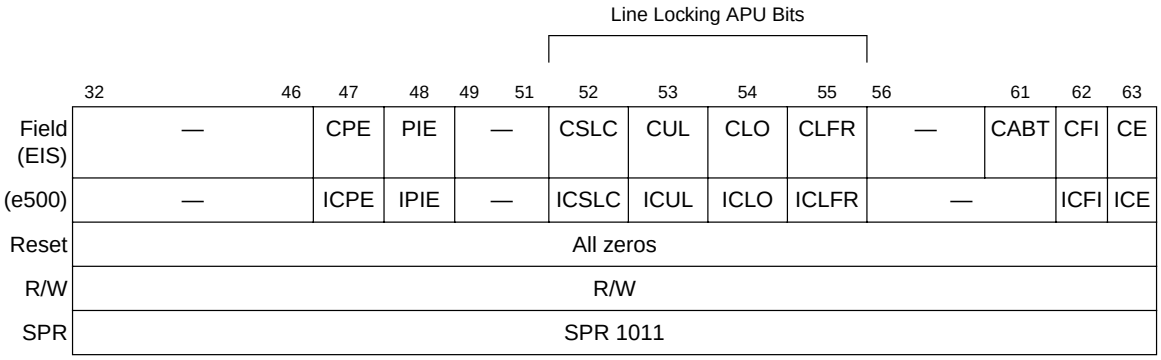


Figure 6-32. L1 Cache Control and Status Register 1 (L1CSR1)

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. 111 Indicates 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

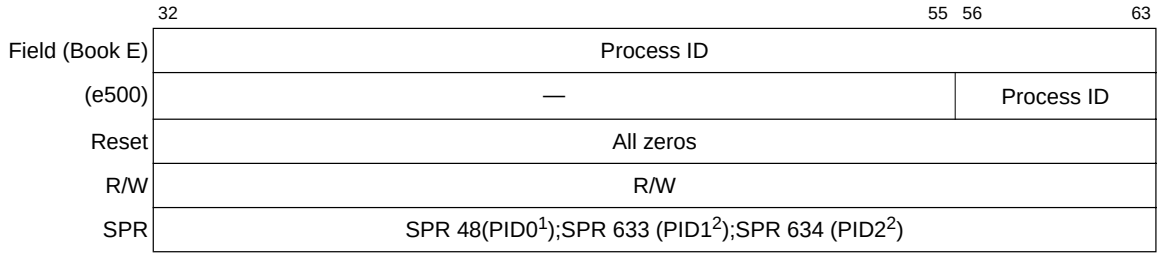
Bits	Name	Description
32–38	—	Reserved, should be cleared.
39–40	ICBSIZ	Instruction cache block size. 00 Indicates block size of 32 bytes
41–42	ICREPL	Instruction cache replacement policy. 01 Indicates pseudo-LRU policy
43	ICLA	Instruction cache locking available. 1 indicates available.
44	ICPA	Instruction 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)



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)

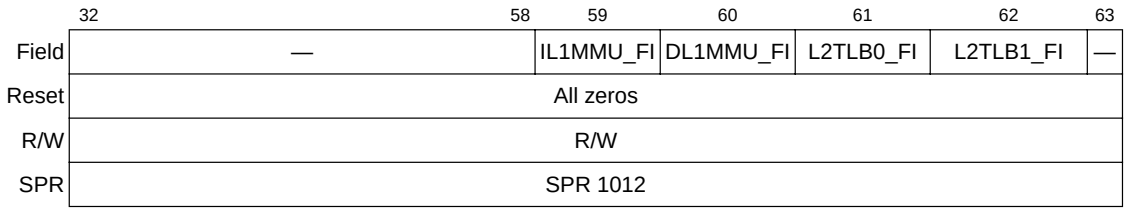


Figure 6-36. MMU Control and Status Register 0 (MMUCSR0)

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 (TLB*n*CFG)

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 (TLB*0*CFG)

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)

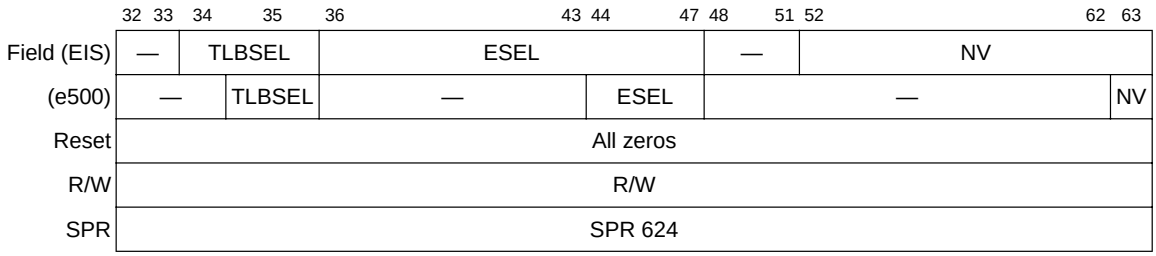


Figure 6-40. MAS Register 0 (MAS0) Format

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)

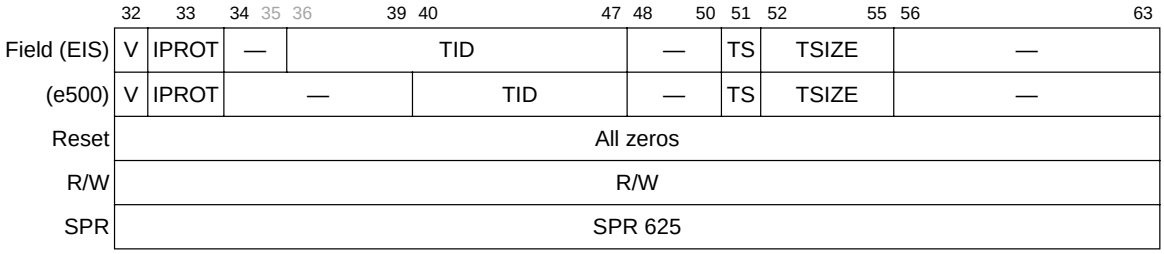


Figure 6-41. MAS Register 1 (MAS1) Format

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 endianess. 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 … 51	52 53	54 … 57	58	59	60	61	62	63
Field	RPN	—	U0–U3	UX	SX	UW	SW	UR	SR
Reset	All zeros								
R/W	R/W								
SPR	SPR 627								

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)

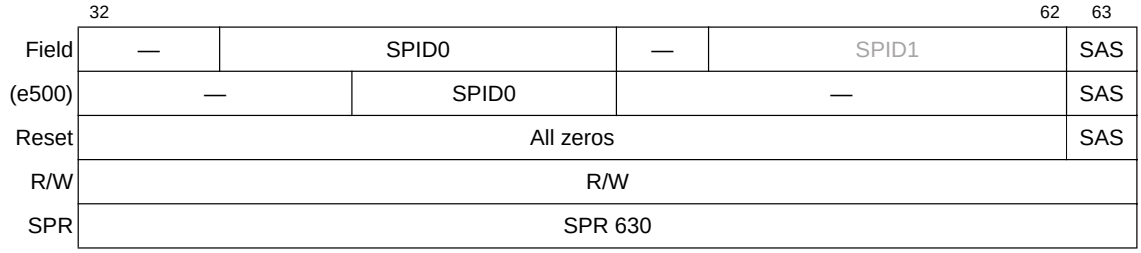


Figure 6-45. MAS Register 6 (MAS6)

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)

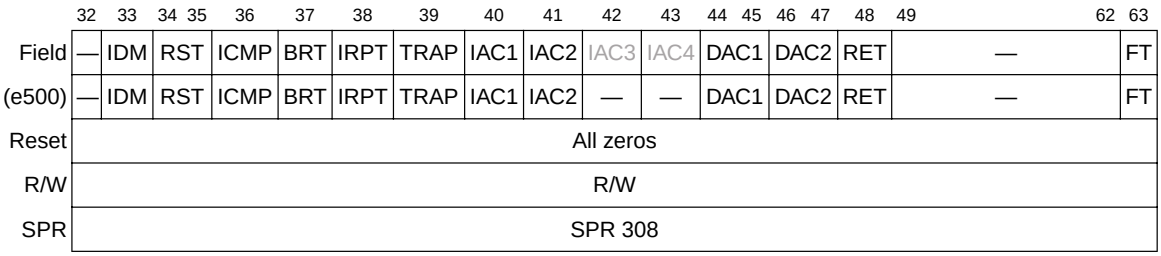


Figure 6-46. Debug Control Register 0 (DBCR0)

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: 0*x* Default (No action) 1*x* 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 $\overline{\text{UDE}}$ signal (level sensitive, active low) is asserted, DBSR[UDE] is affected as follows: MSR[DE] DBCR0[IDM] 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)

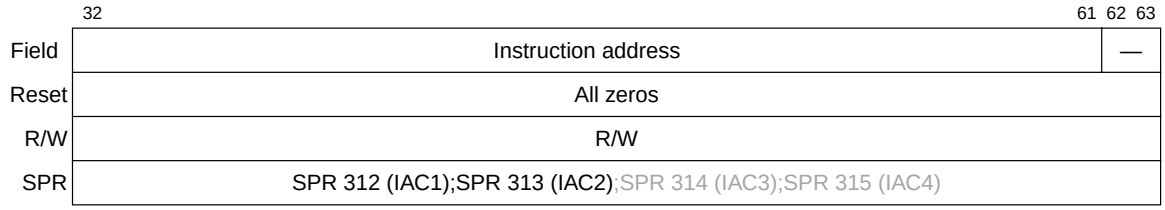


Figure 6-50. Instruction Address Compare Registers (IAC1–IAC2)

6.13.4 Data Address Compare Registers (DAC1–DAC2)

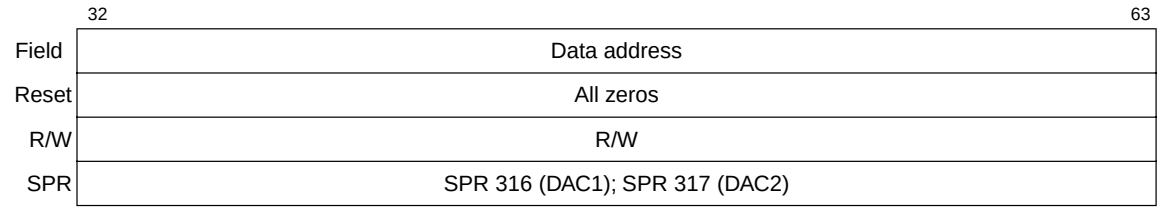


Figure 6-51. Data Address Compare Registers (DAC1–DAC2)

6.14 Signal Processing Engine (SPE) APU Registers

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

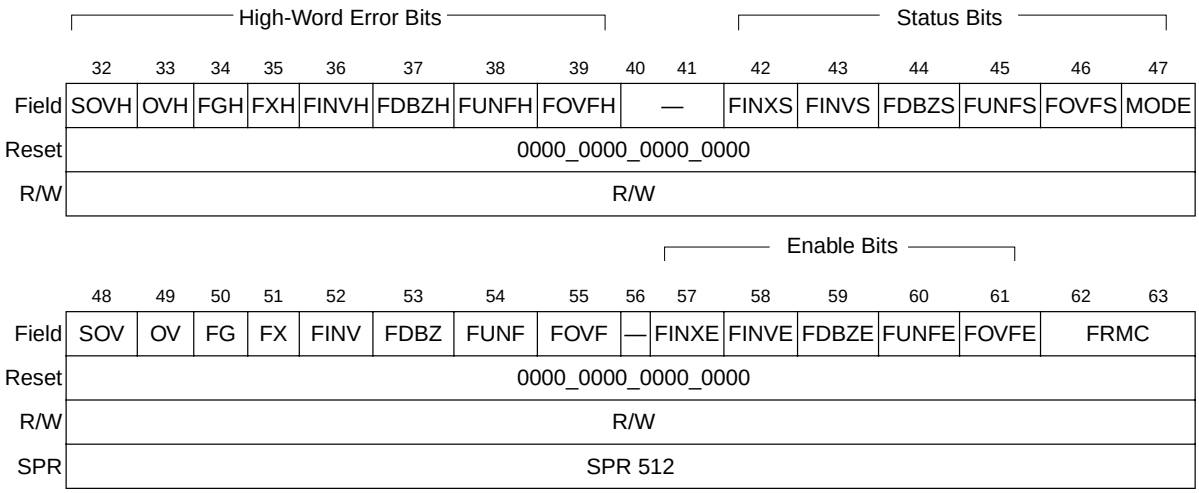


Figure 6-52. Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)

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 | FGH | FXH | FG | 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 of the dividend is non-zero and the divisor is zero.
54	FUNF	Embedded floating-point underflow error
55	FOVF	Embedded floating-point overflow error
56	—	Reserved, should be cleared
57	FINXE	Embedded floating-point inexact enable
58	FINVE	Embedded floating-point invalid operation 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)

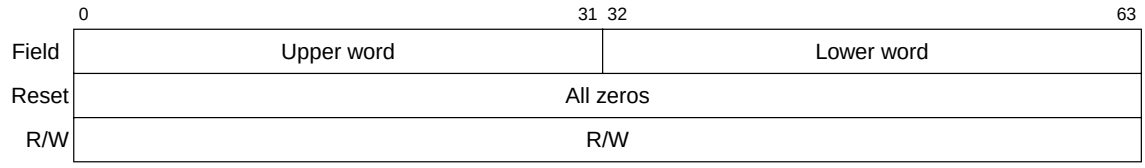


Figure 6-53. Accumulator (ACC)

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)

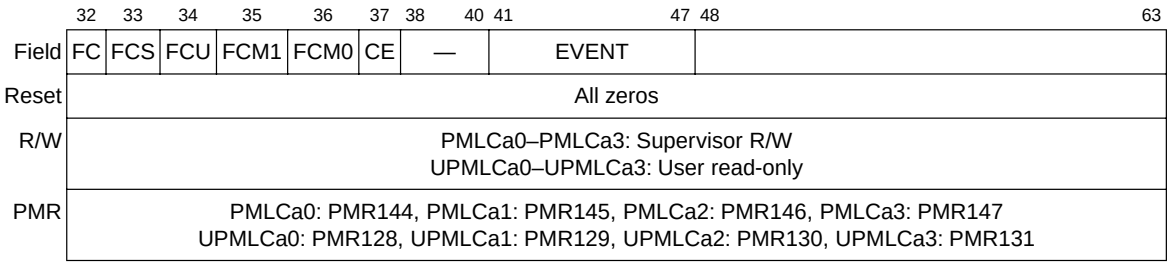


Figure 6-55. Local Control A Registers (PMLCa0–PMLCa3)
User Local Control A Registers (UPMLCa0–UPMLCa3)

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

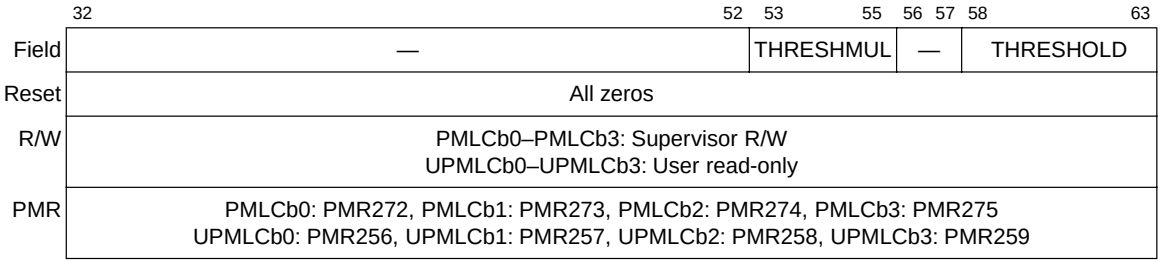


Figure 6-56. Local Control B Registers (PMLCb0–PMLCb3)
User Local Control B Registers (UPMLCb0–UPMLCb3)

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 (PMLCb*n*[THRESHOLD] * 1) 001 Threshold field is multiplied by 2 (PMLCb*n*[THRESHOLD] * 2) 010 Threshold field is multiplied by 4 (PMLCb*n*[THRESHOLD] * 4) 011 Threshold field is multiplied by 8 (PMLCb*n*[THRESHOLD] * 8) 100 Threshold field is multiplied by 16 (PMLCb*n*[THRESHOLD] * 16) 101 Threshold field is multiplied by 32 (PMLCb*n*[THRESHOLD] * 32) 110 Threshold field is multiplied by 64 (PMLCb*n*[THRESHOLD] * 64) 111 Threshold field is multiplied by 128 (PMLCb*n*[THRESHOLD] * 128)
56–57	—	Reserved, should be cleared.
58–63	THRESHOLD	Threshold. Only events that exceed the THRESHOLD value are counted. 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 MPC8560 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.

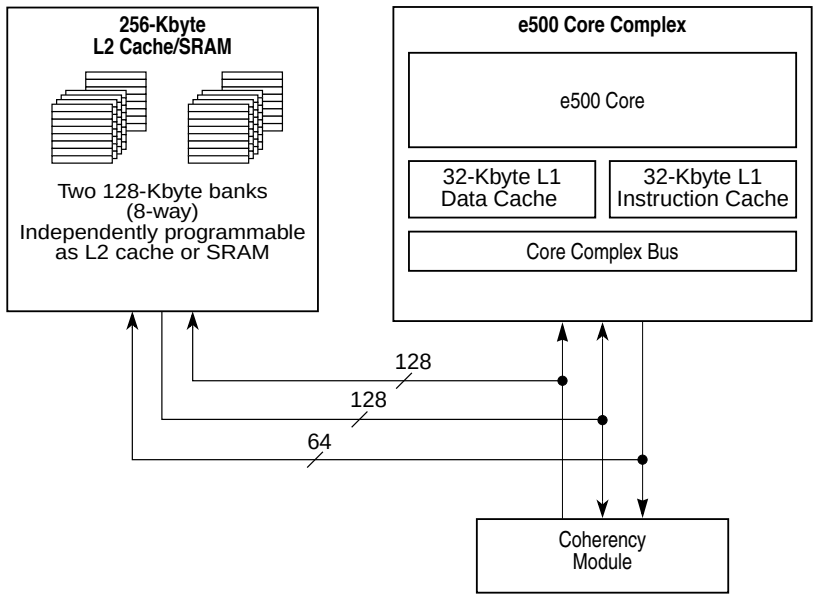


Figure 7-1. L2 Cache/SRAM Configuration

The L2 cache can operate in the following modes:

- Full cache mode (256-Kbyte cache).

- 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.

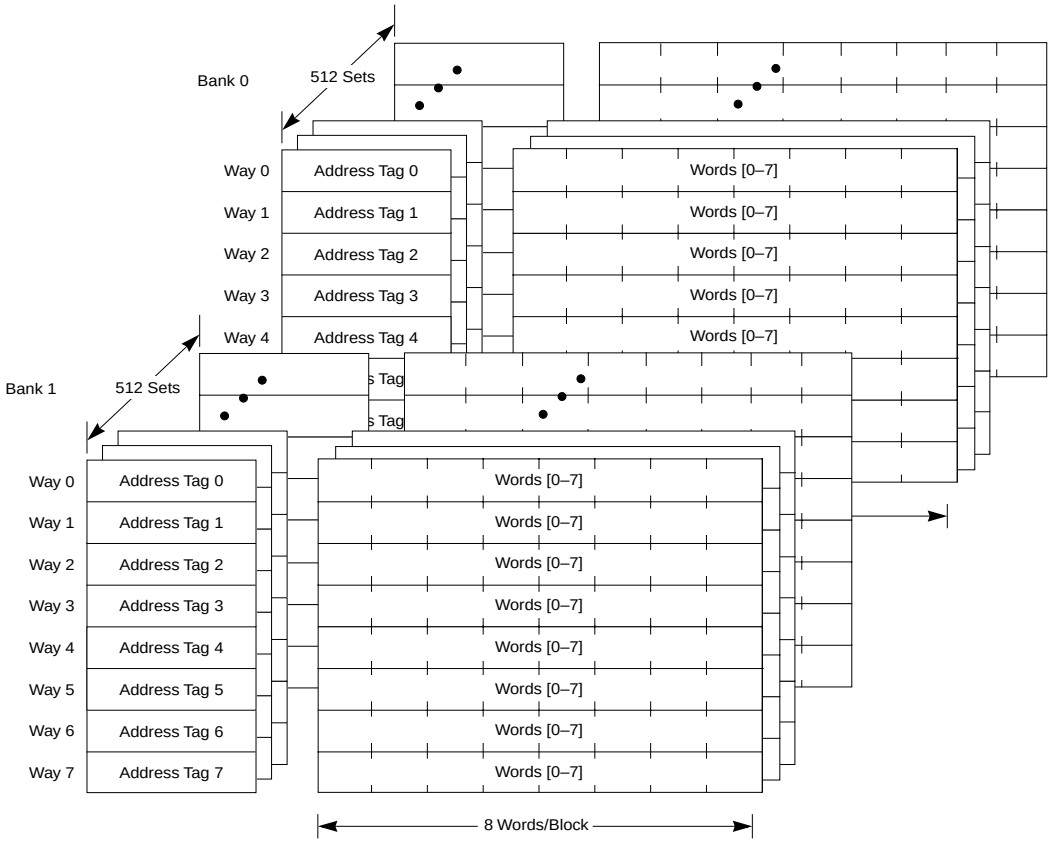


Figure 7-2. Cache Organization

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.

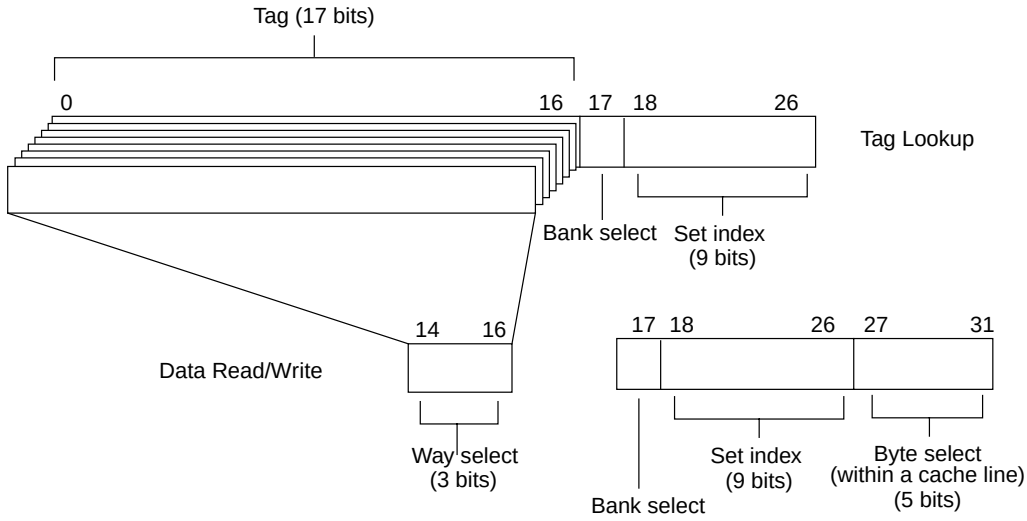


Figure 7-3. 256-Kbyte L2 Cache Address Configuration—Full Cache Mode

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.

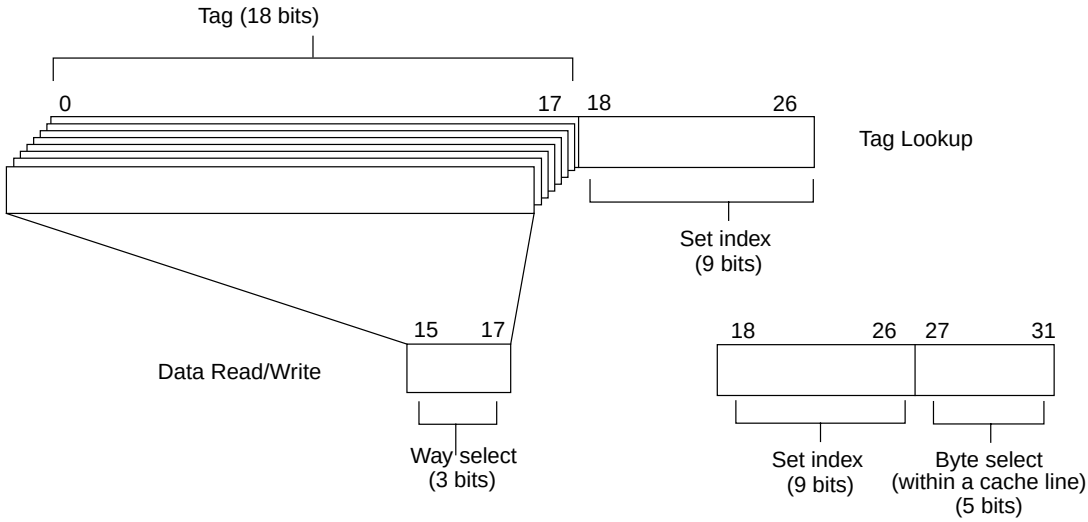


Figure 7-4. 128-Kbyte L2 Cache Address Configuration—Half SRAM, Half Cache Mode

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.

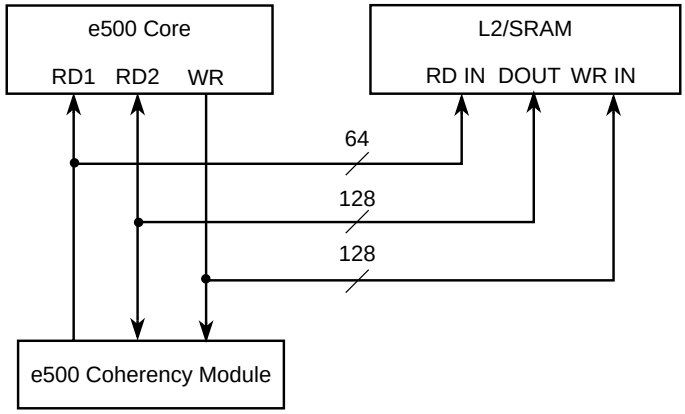


Figure 7-5. Data Bus Connection of CCB

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

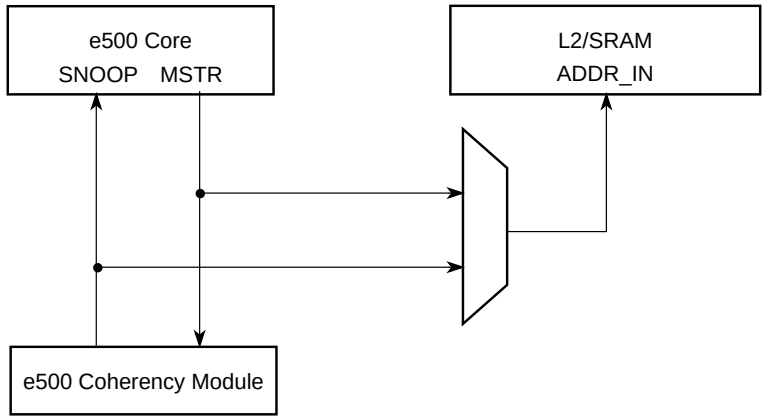


Figure 7-6. Address Bus Connection of CCB

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**

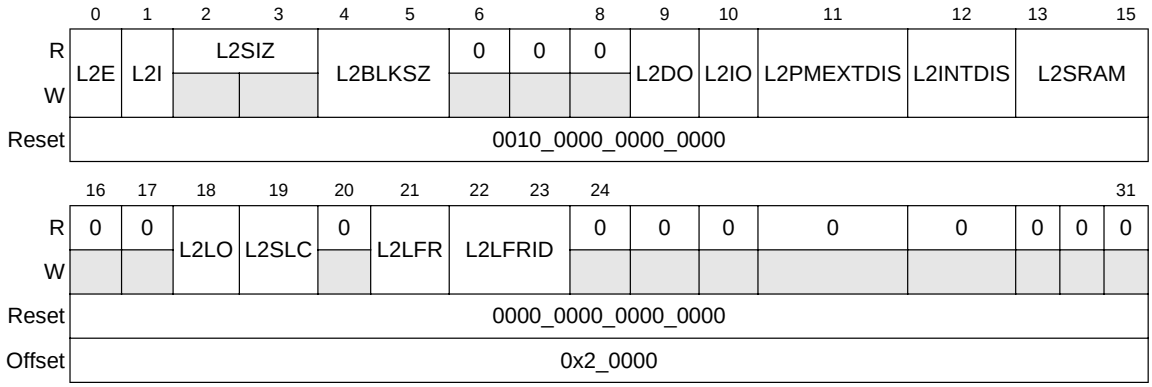


Figure 7-7. L2 Control Register (L2CTL)

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. L2SIZ = L2BLKSIZ (1 block): 000 Block 0 = Cache 001 Block 0 = SRAM0 01x Reserved 1xx Reserved L2SIZ = L2BLKSIZ *2 (2 blocks): 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 MPC8560 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.

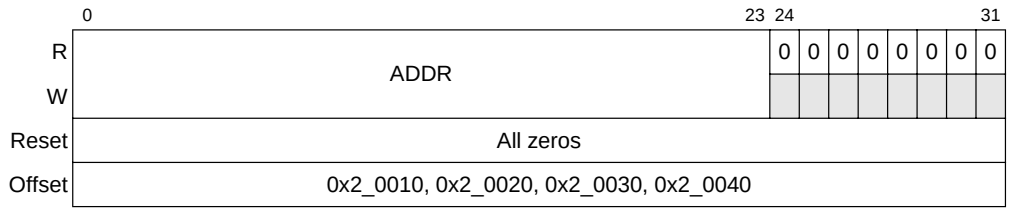


Figure 7-8. Cache External Write Address Registers (L2CEWAR*n*)

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–L2CEWCR03)

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.

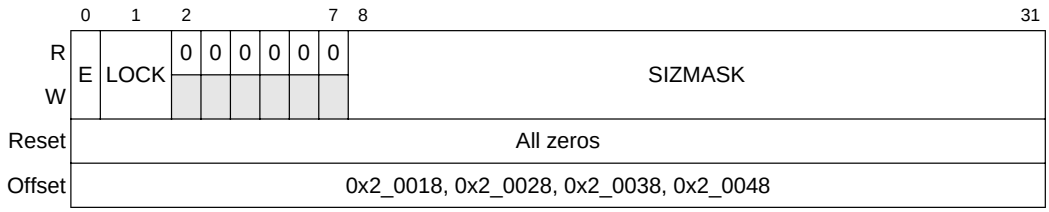


Figure 7-9. Cache External Write Control Registers (L2CEWCR0–L2CEWCR3)

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 L2CEWARx/L2CEWCRx is allocated or updated inthe 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 L2CEWARx/L2CEWCRx 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.

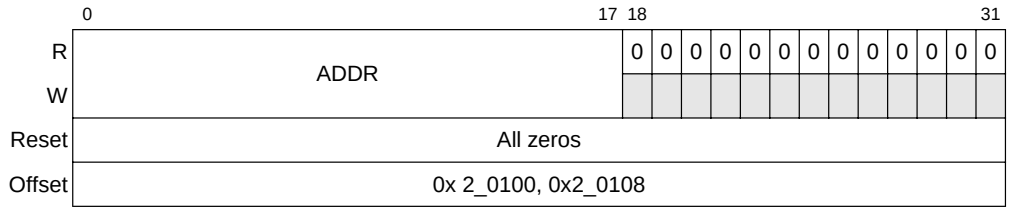


Figure 7-10. L2 Memory-Mapped SRAM Base Address Registers (L2SRBAR*n*)

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 MPC8560 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
dcbtls_L2 A     | forces the line from the L1 and allocates the line in the L2
lwz A
```

Data or tag errors are injected into the line, 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).

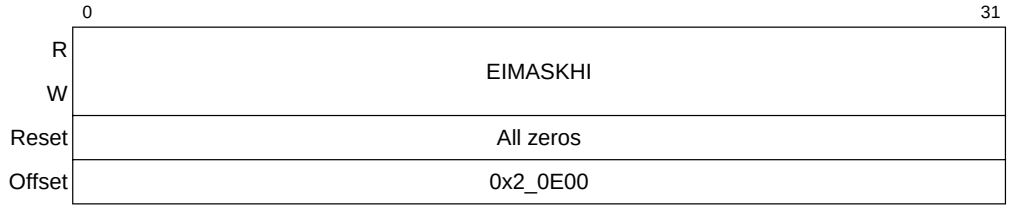


Figure 7-11. L2 Error Injection Mask High Register (L2ERRINJHI)

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).

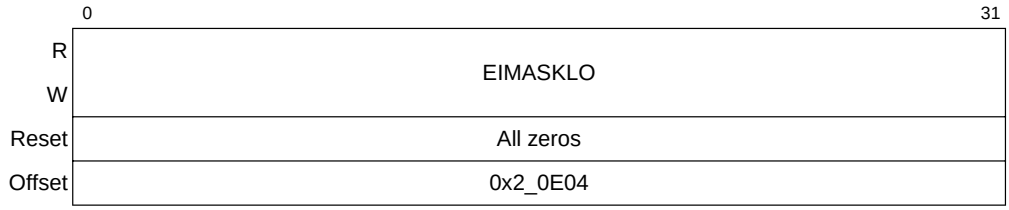


Figure 7-12. L2 Error Injection Mask Low Register (L2ERRINJLO)

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).

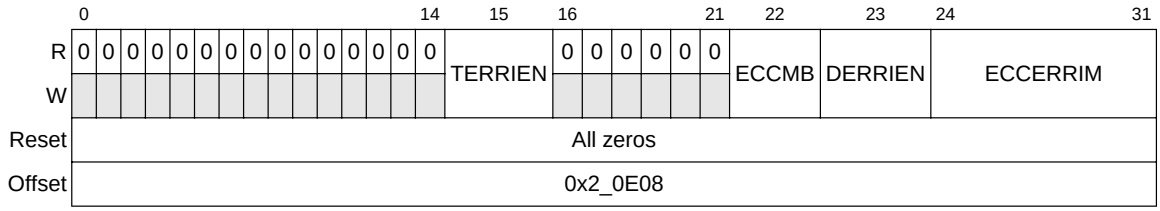


Figure 7-13. L2 Error Injection Mask Control Register (L2ERRINJCTL)

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).

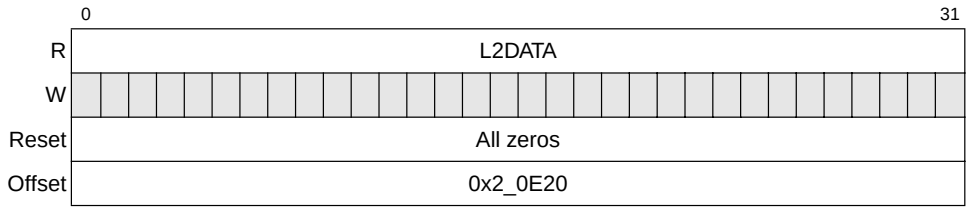


Figure 7-14. L2 Error Capture Data High Register (L2CAPTDATAHI)

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).

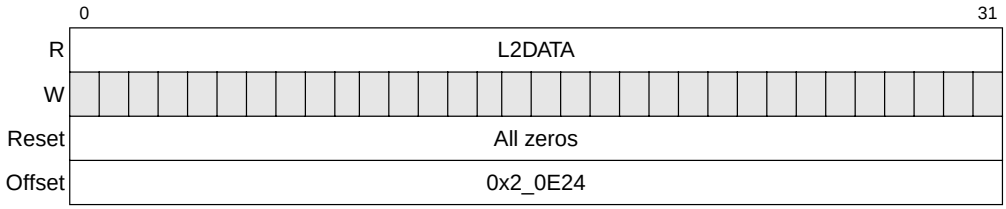


Figure 7-15. L2 Error Capture Data Low Register (L2CAPTDATALO)

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).

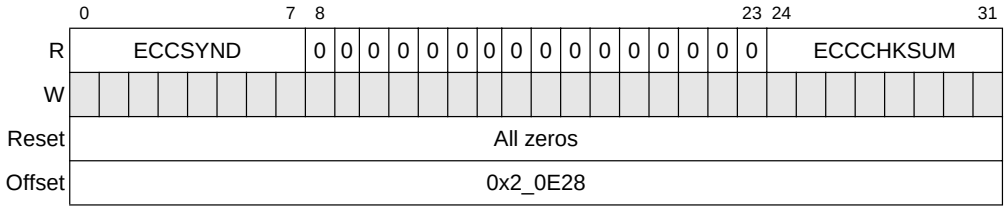


Figure 7-16. L2 Error Syndrome Register (L2CAPTECC)

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
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	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).

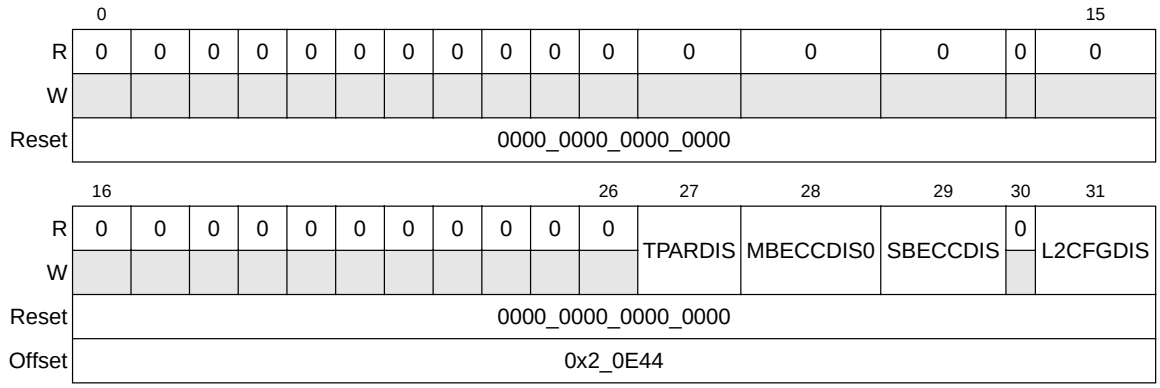


Figure 7-18. L2 Error Disable Register (L2ERRDIS)

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.

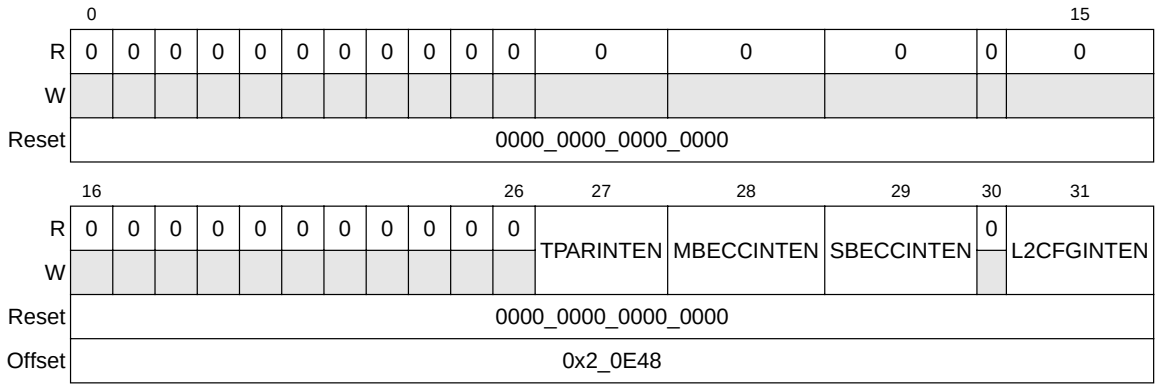


Figure 7-19. L2 Error Interrupt Enable Register (L2ERRINTEN)

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).

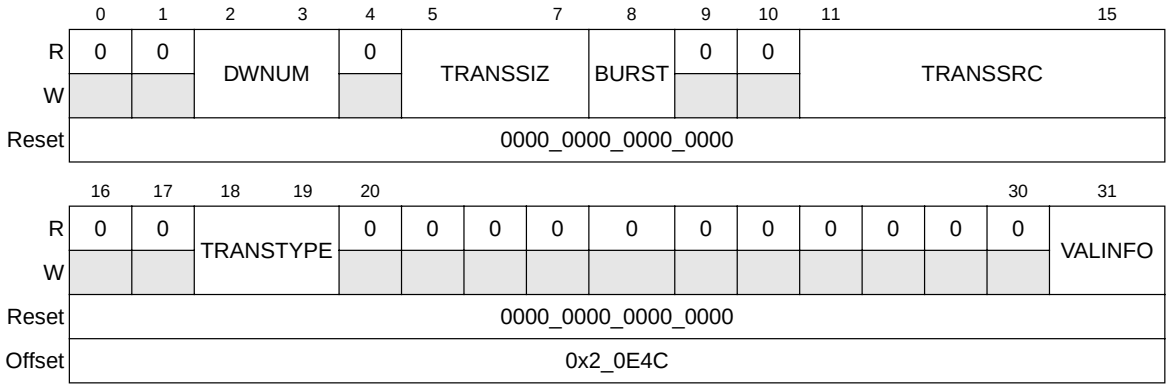


Figure 7-20. L2 Error Attributes Capture Register (L2ERRATTR)

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 00000External (system logic) 10000Processor (inst) 10001Processor (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).

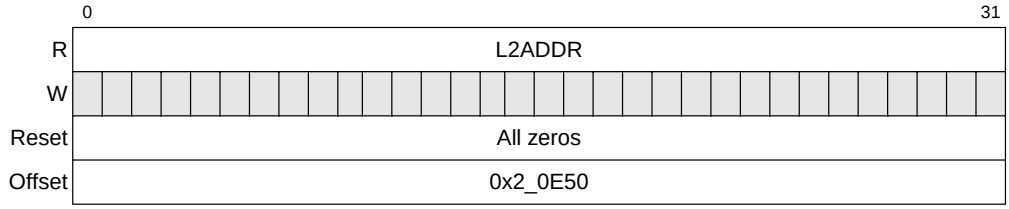


Figure 7-21. L2 Error Address Error Capture Register (L2ERRADDR)

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).

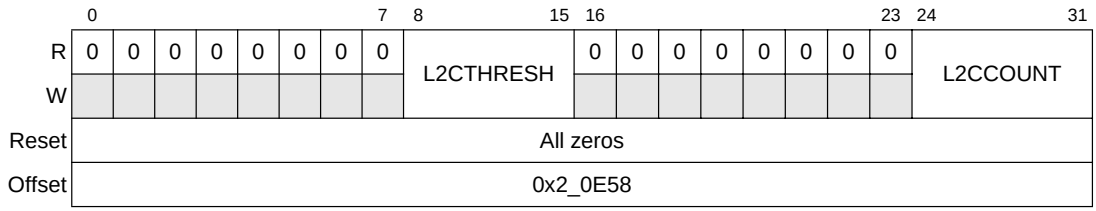


Figure 7-22. L2 Error Control Register (L2ERRCTL)

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>
CCB addr bus load 1		BG	TS		AACK	HIT DATA-COMING					
CCB addr bus load 2				BG	TS		AACK	HIT DATA-COMING			
CCB data bus load 1							DATA	DATA			
CCB data bus load 2									DATA	DATA	

7.5 SRAM Coherency

This section explains the rules of cache and memory-mapped SRAM coherency. The term snoop transaction refers to transactions initiated by the MPC8560 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.

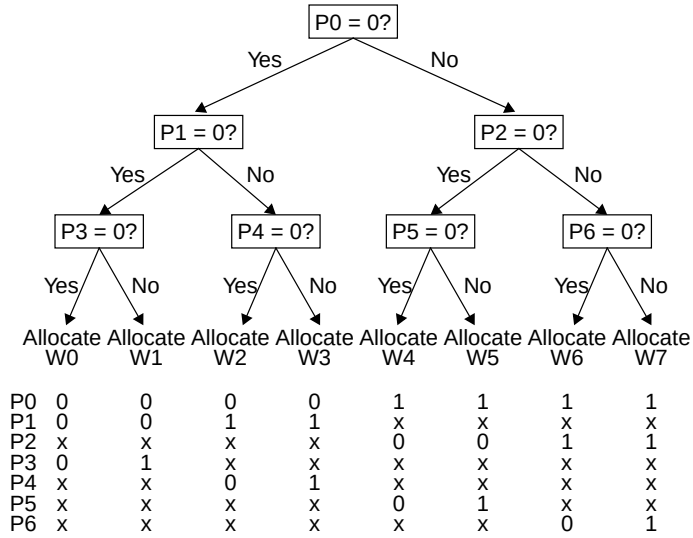


Figure 7-23. L2 Cache Line Replacement Algorithm

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: Li = Di | Ii, i=0..7 (Di = data lock, Ii = instruction lock)
```

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

```
P0_eff = f(P0,L0,L1,L2,L3,L4,L5,L6,L7) = (L0 & L1 & L2 & L3) | (P0 & ~(L4 & L5 & L6 & L7))
P1_eff = f(P1,L0,L1,L2,L3) = (L0 & L1) | (P1 & ~(L2 & L3))
P2_eff = f(P2,L4,L5,L6,L7) = (L4 & L5) | (P2 & ~(L6 & L7))
P3_eff = f(P3,L0,L1) = L0 | (P3 & ~L1)
P4_eff = f(P4,L2,L3) = L2 | (P4 & ~L3)
P5_eff = f(P5,L4,L5) = L4 | (P5 & ~L5)
P6_eff = f(P6,L6,L7) = L6 | (P6 & ~L7)
```

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

Table 7-20. PLRU-Based Victim Selection Mechanism

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

7.8 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 icbtls_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
icbtls_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) dcbtls_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) dcbtls_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 dcbtstls_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	
dcbtls_L2 dcbtstls_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
dcblc_L2 icblc_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
dcbtls_L2 icbtls_L2 dcbtstls_L2		I	No	EL	I	An icbtls_L2 that hits modified in L1 cannot be distinguished from dcbtls_L2 and sets the L2 dlock bit. If software shares cache lines between instructions and data and wishes to set 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 MPC8560 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 MPC8560.

  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 MPC8560.
- Chapter 9, “DDR Memory Controller” describes the DDR SDRAM memory controller of the MPC8560. 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 overide, crawl mode, and ECC error injection support rapid system debug.
- Chapter 10, “Programmable Interrupt Controller” describes the embedded programmable interrupt controller (PIC) of the MPC8560. 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 I^2C) bus controller of the MPC8560. 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 MPC8560 powers up in boot sequencer mode which allows the I^2C controller to initialize configuration registers.

- Chapter 12, “Local Bus Controller” describes the local bus controller of the MPC8560. 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 13, “Three-Speed Ethernet Controller” describes the two triple-speed Ethernet controllers on the MPC8560. 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 MPC8560 memory coherency module.
- Chapter 14, “DMA Controller,” describes the four-channel general-purpose DMA controller of the MPC8560. 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 15, “PCI Bus Interface,” describes the PCI/PCI-X controller of the MPC8560.
- Chapter 16, “RapidIO Interface,” describes the RapidIO controller of the MPC8560.

Chapter 8
e500 Coherency Module

8.1 Introduction

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

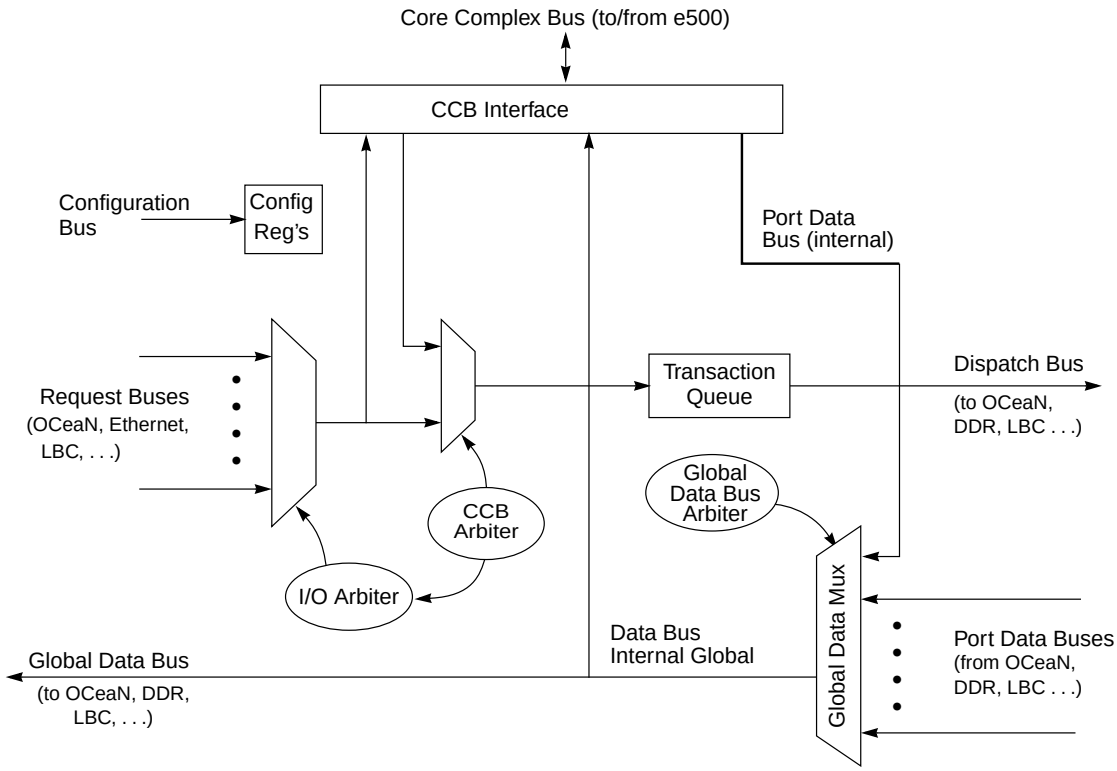


Figure 8-1. e500 Coherency Module Block Diagram

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 the CPM and the 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 0x000–0xFFF 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)

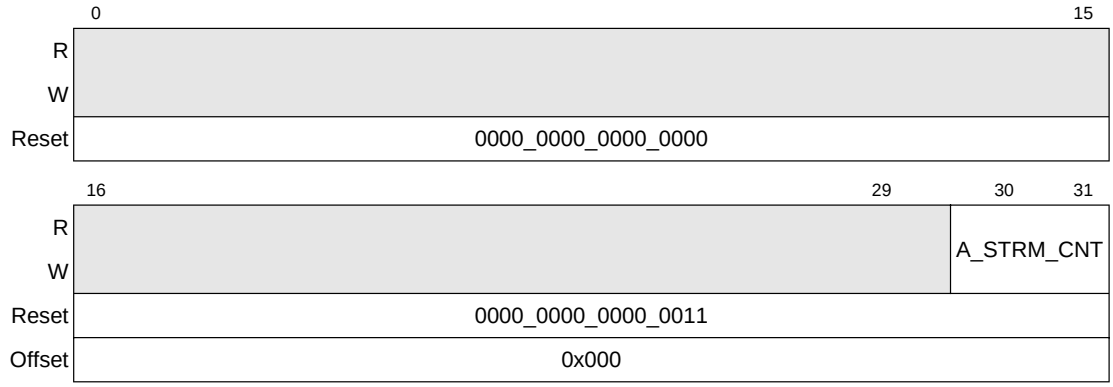


Figure 8-2. ECM CCB Address Configuration Register (EEBACR)

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)

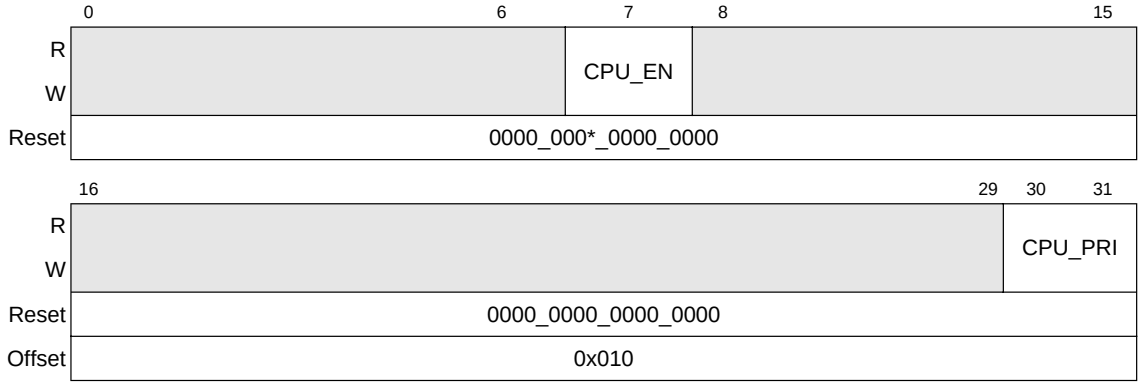


Figure 8-3. ECM CCB Port Configuration Register (EEBPCR)

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)

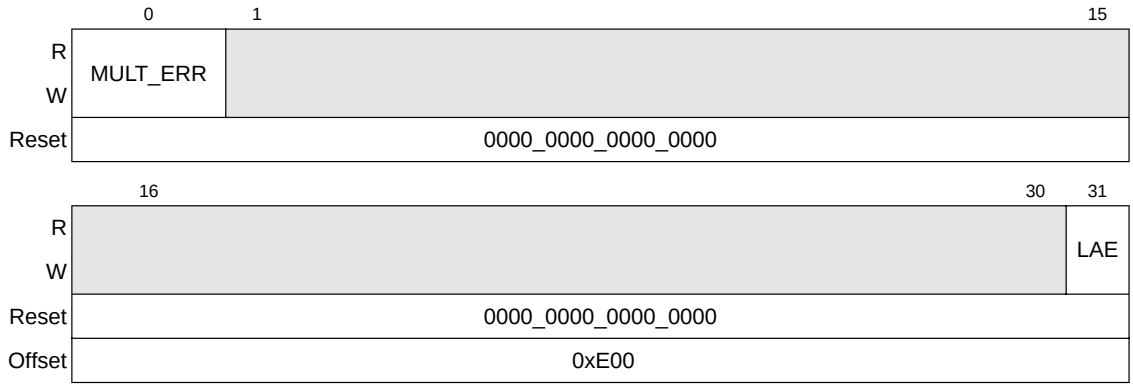


Figure 8-4. ECM Error Detect Register (EEDR)

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 targetting 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.

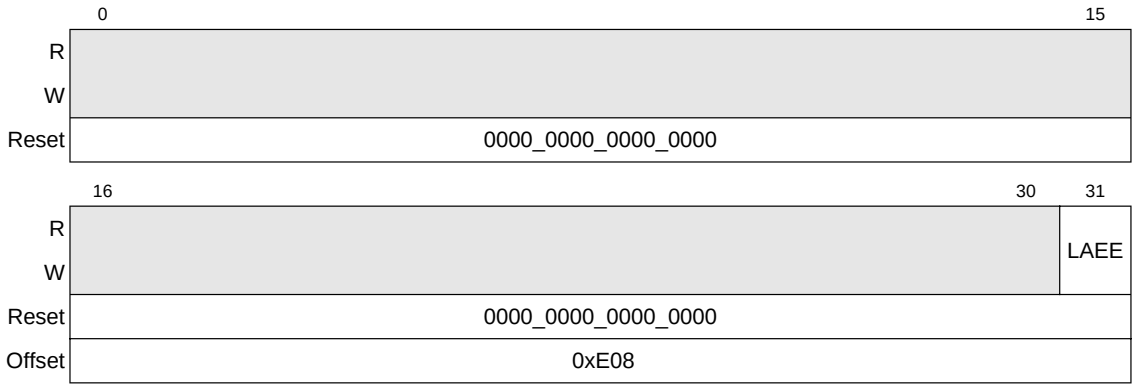


Figure 8-5. ECM Error Enable Register (EEER)

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)

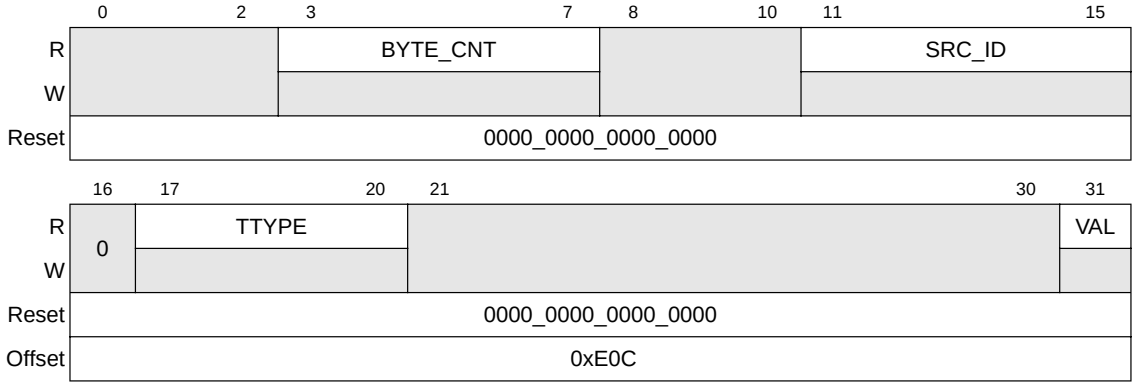


Figure 8-6. ECM Error Attributes Capture Register (EEATR)

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. Iidentifies 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 CPM 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	TTYPE	Transaction type. Defined as follows: 0000 Write 0001 Reserved 0010 Write with allocate 0011 Write with allocate with lock 0100 Address only transaction 0101 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)

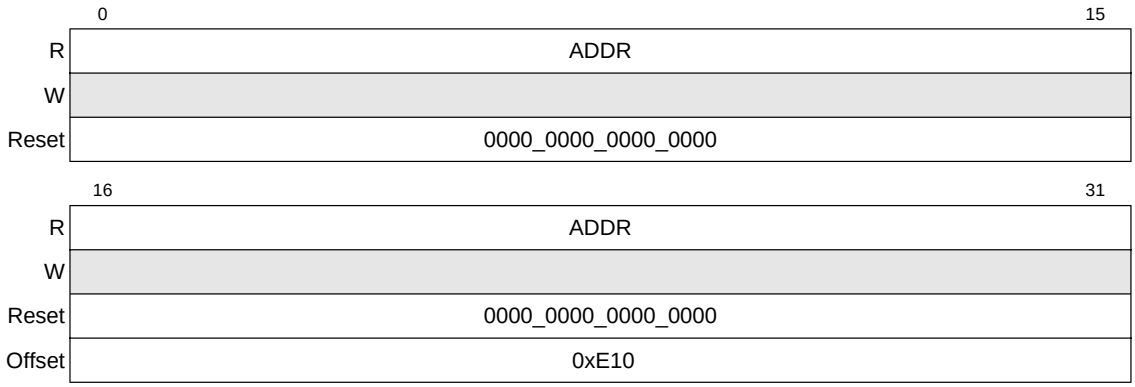


Figure 8-7. ECM Error Address Capture Register (EEADR)

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 MPC8560, 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 MPC8560, 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 initiallization. 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 overide, 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.

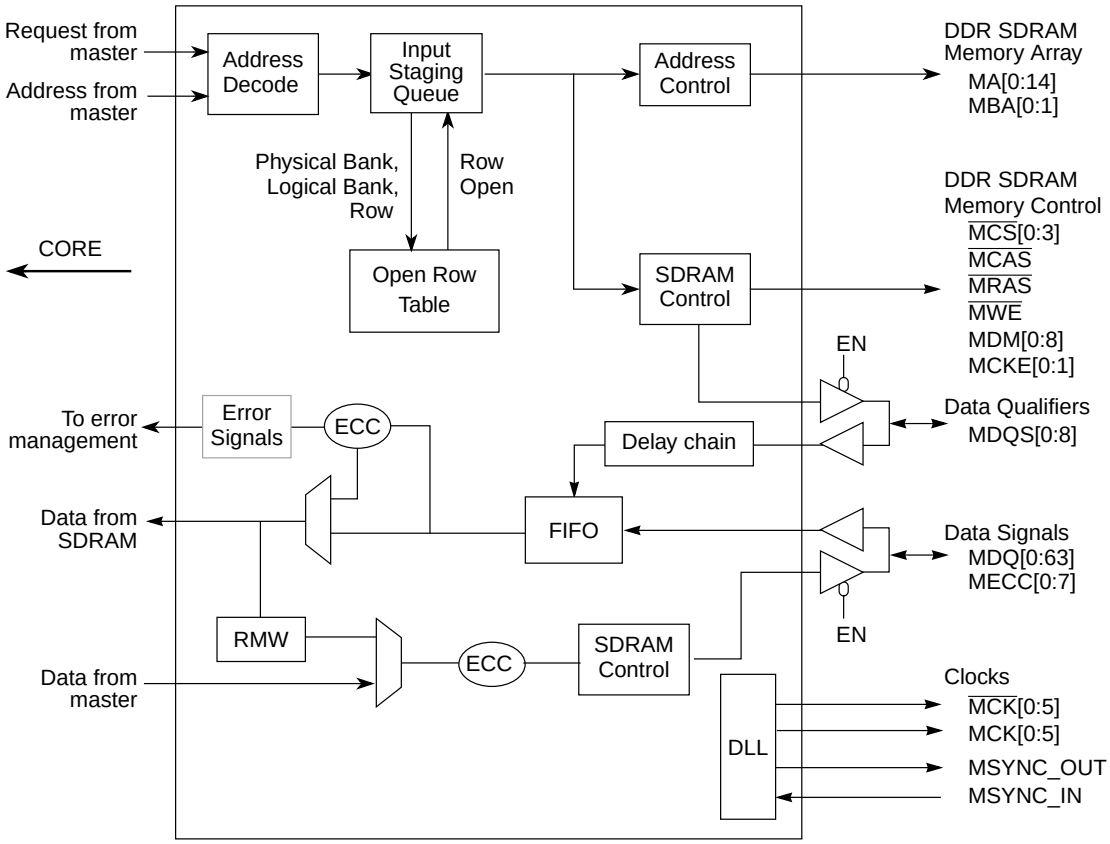


Figure 9-1. DDR Memory Controller Simplified Block Diagram

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 $\overline{\text{AS}}$ (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 *MPC8560 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 TIMING_CFG_1 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. SeeTable 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 $\overline{\text{MCSx}}$ 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], $\overline{\text{MCK}}$[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 MPC8560This 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 MPC8560. 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 loopSee 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 (CS*n*_BNDS)

The chip select bounds registers (CS*n*_BNDS) shown in Figure 9-2 define the starting address of the memory space that corresponds to the individual chip selects.

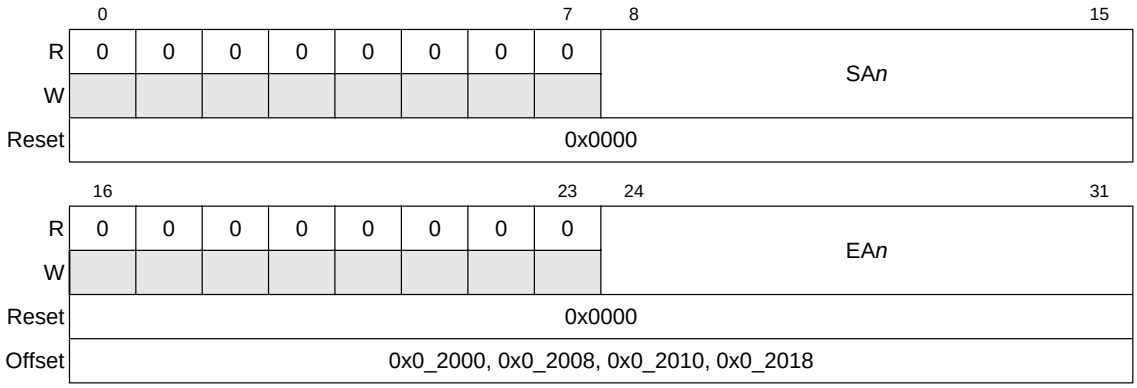


Figure 9-2. Chip Select Bounds Registers (CS*n*_BNDS)

Table 9-7 describes the CS*n*_BNDS register fields.

Table 9-7. CS*n*_BNDS Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	SA*n*	Starting address for chip select (bank) *n.* This value is compared against the most significant 8 bits of the address.
16–23	—	Reserved
24–31	EA*n*	Ending address for chip select (bank) *n.* This value is compared against the most significant 8 bits of the address.

9.3.1.2 Chip Select Configuration (CS*n*_CONFIG)

The chip select configuration (CS*n*_CONFIG) registers shown in Figure 9-3 enable the DDR chip selects and set the number of row and column bits used for each chip select.

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 CS*n*_CONFIG[ROW_BITS_CS_*n*] and CS*n*_CONFIG[COL_BITS_CS_*n*] the user should take great care to set these values correctly.

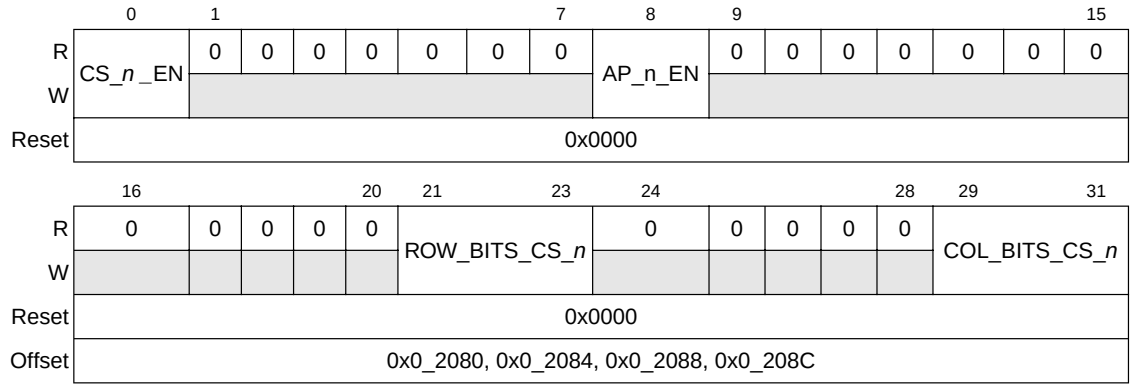


Figure 9-3. Chip Select Configuration Register (CS*n*_CONFIG)

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 *n* enable 0 Chip select *n* is not active 1 Chip select *n* is active and assumes the state set in CSn_BNDS.
1–7	—	Reserved
8	AP_n_EN	Chip select *n* auto precharge enable 0 Chip select *n* will only be auto precharged if global auto precharge mode is enabled ('BSTOPRE' = 0). 1 Chip select *n* 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 *n* 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 *n* 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.

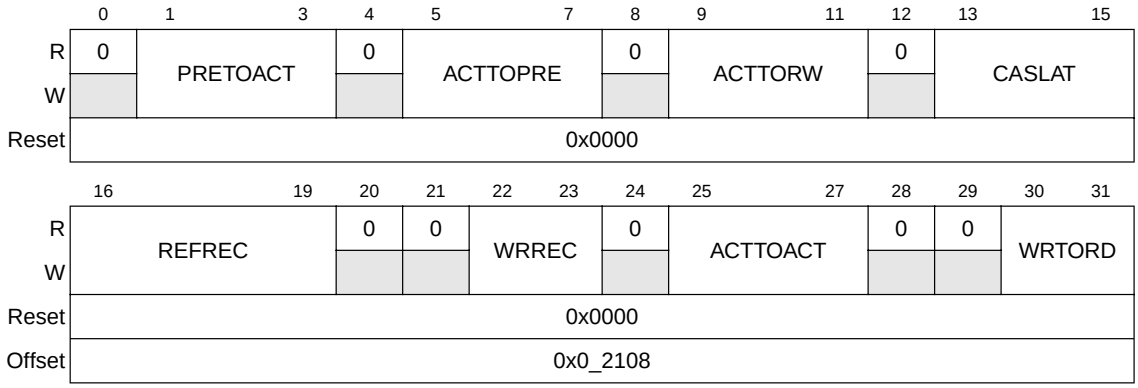


Figure 9-4. DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)

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_{rcd}). 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	$\overline{\text{MCAS}}$ 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.

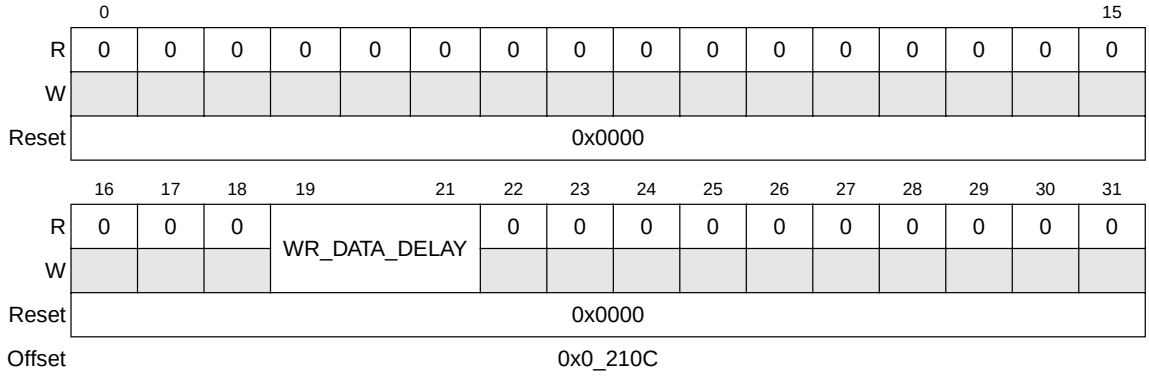


Figure 9-5. DDR SDRAM Timing Configuration 2 Register (TIMING_CFG_2)

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.

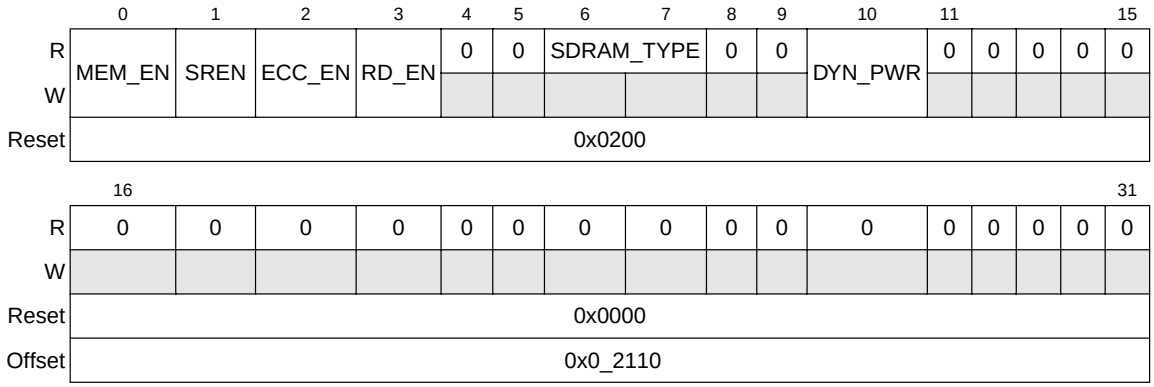


Figure 9-6. DDR SDRAM Control Configuration Register (DDR_SDRAM_CFG)

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.

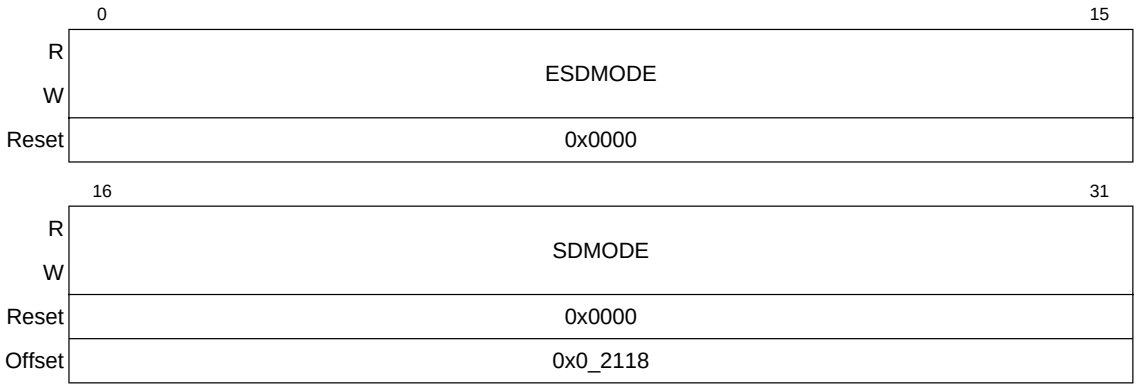


Figure 9-7. DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)

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 intialization 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.

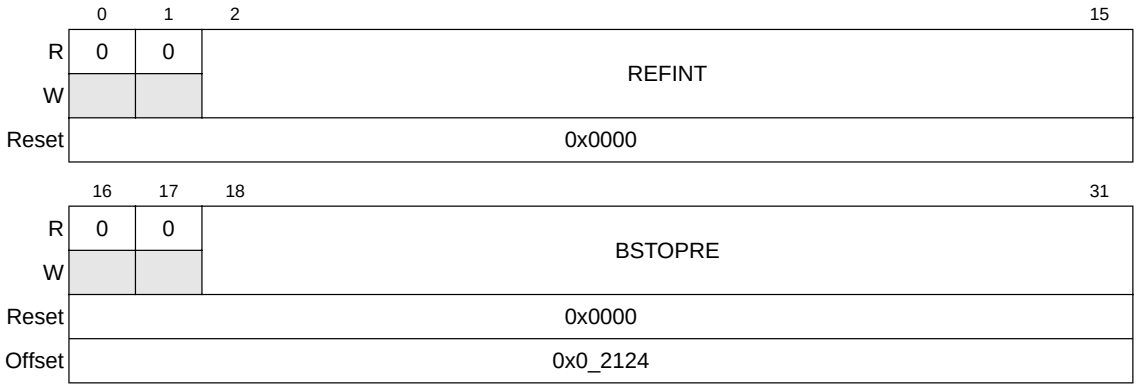


Figure 9-8. DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)

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.

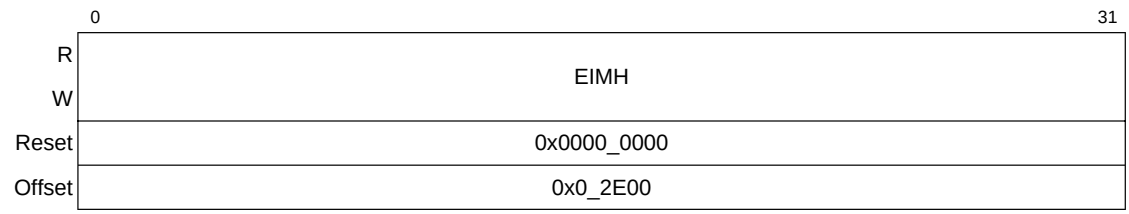


Figure 9-9. Memory Data Path Error Injection Mask High Register (DATA_ERR_INJECT_HI)

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.

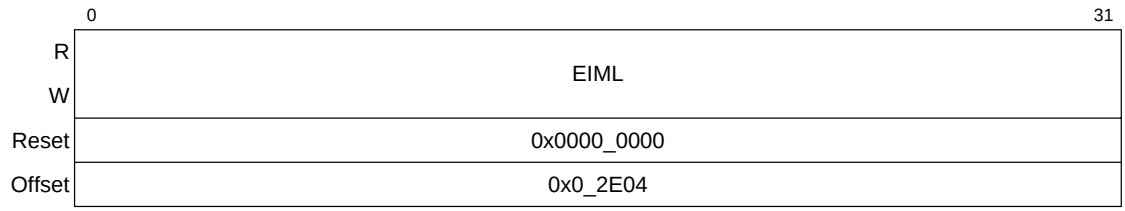


Figure 9-10. Memory Data Path Error Injection Mask Low Register (DATA_ERR_INJECT_LO)

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.

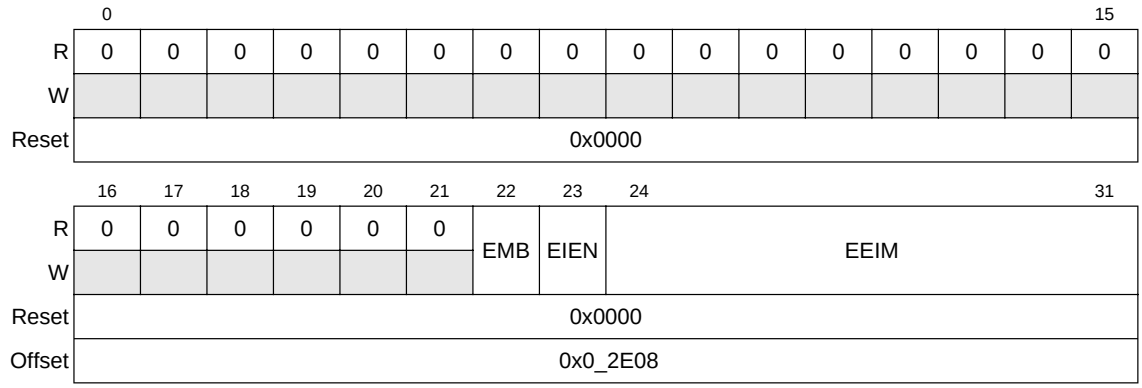


Figure 9-11. Memory Data Path Error Injection Mask ECC Register (ECC_ERR_INJECT)

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.

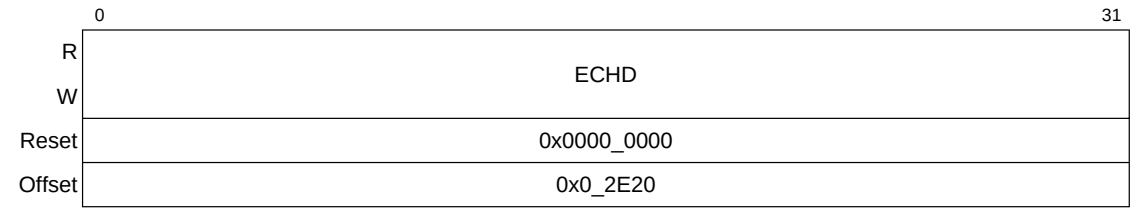


Figure 9-12. Memory Data Path Read Capture High Register (CAPTURE_DATA_HI)

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.

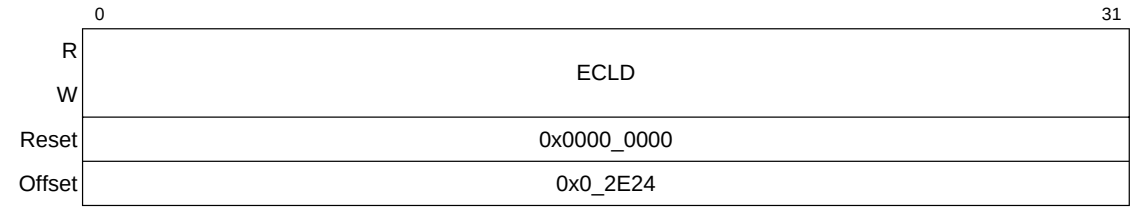


Figure 9-13. Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO)

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.

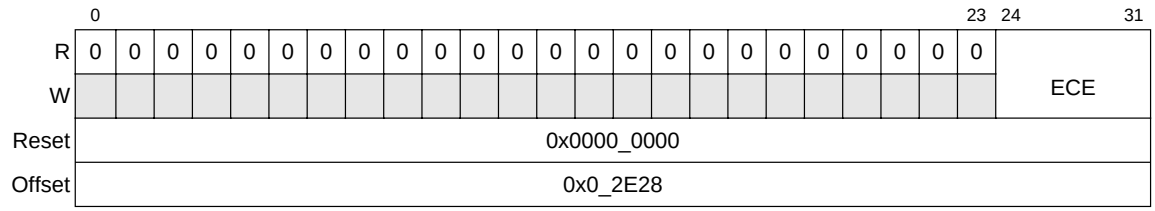


Figure 9-14. Memory Data Path Read Capture ECC Register (CAPTURE_ECC)

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	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.

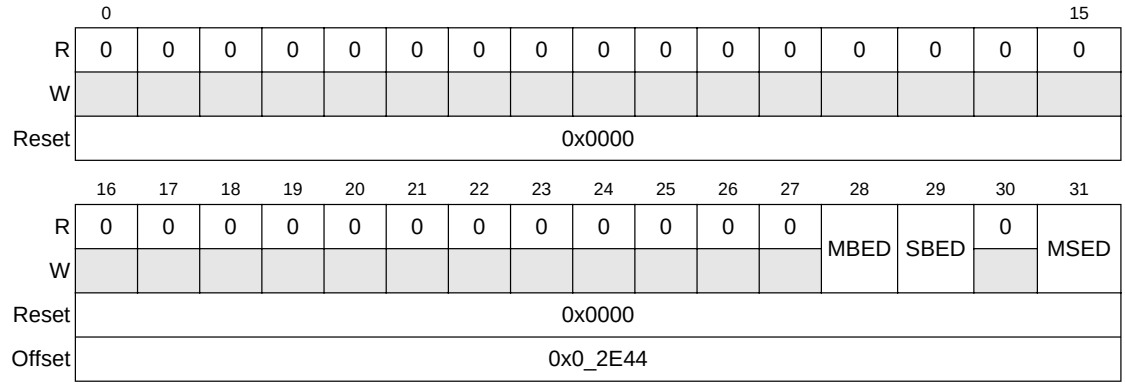


Figure 9-16. Memory Error Disable Register (ERR_DISABLE)

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).

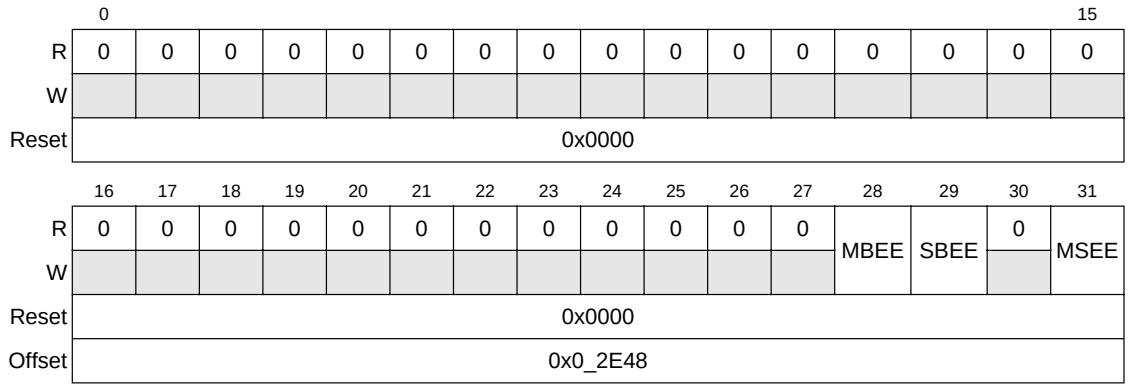


Figure 9-17. Memory Error Interrupt Enable Register (ERR_INT_EN)

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.

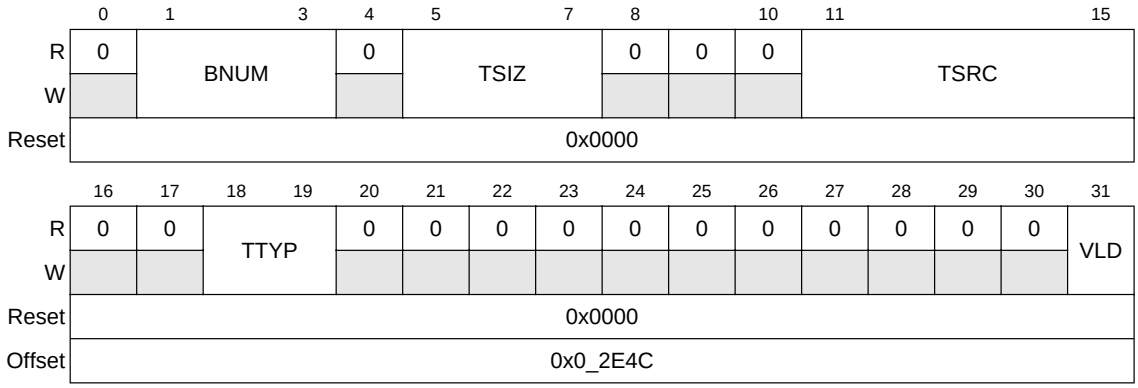


Figure 9-18. Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES)

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 — 00010 — 00011 — 00100 — Local bus 00101 — reserved 00110 — reserved 00111 — reserved 01000 — Configuration space 01001 — 01010 — Boot sequencer 01011 — reserved 01100 — RapidIO 01101 — 01110 — 01111 — 10000 — Processor (inst) 10001 — Processor (data) 10010 — 10011 — 10100 — CPM 10101 — DMA 10110 — 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.

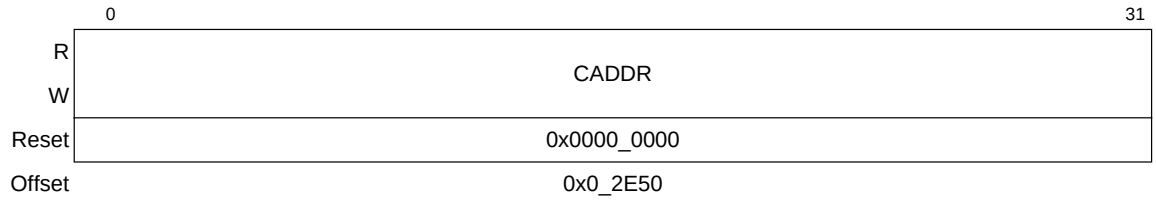


Figure 9-19. Memory Error Address Capture Register (CAPTURE_ADDRESS)

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.

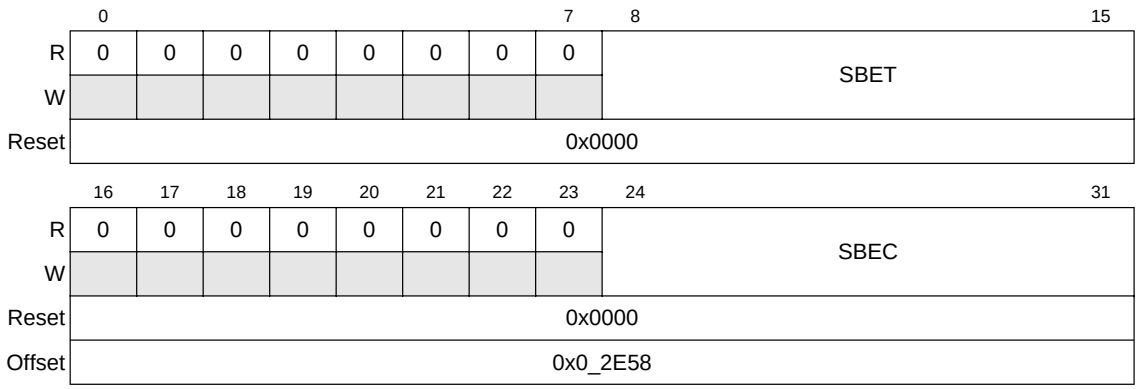


Figure 9-20. Single-Bit ECC Memory Error Management Register (ERR_SBE)

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.

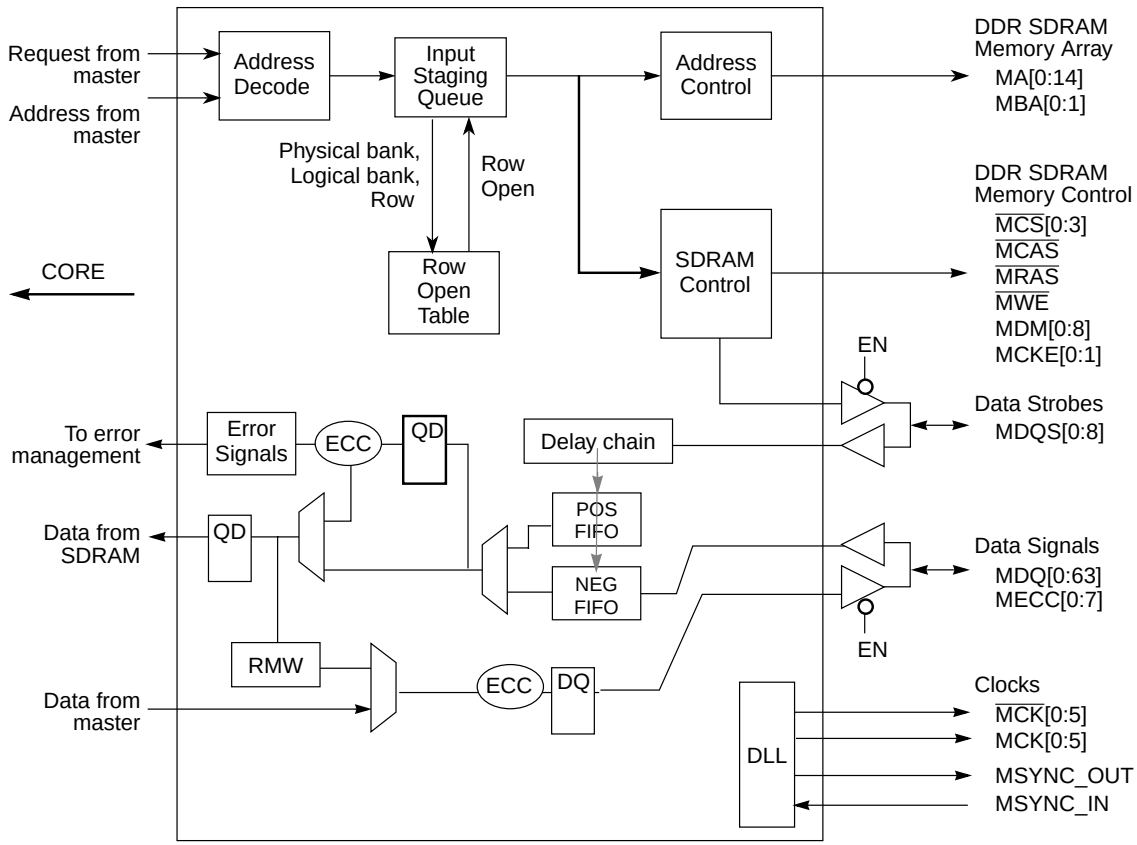


Figure 9-21. DDR Memory Controller Block Diagram

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.

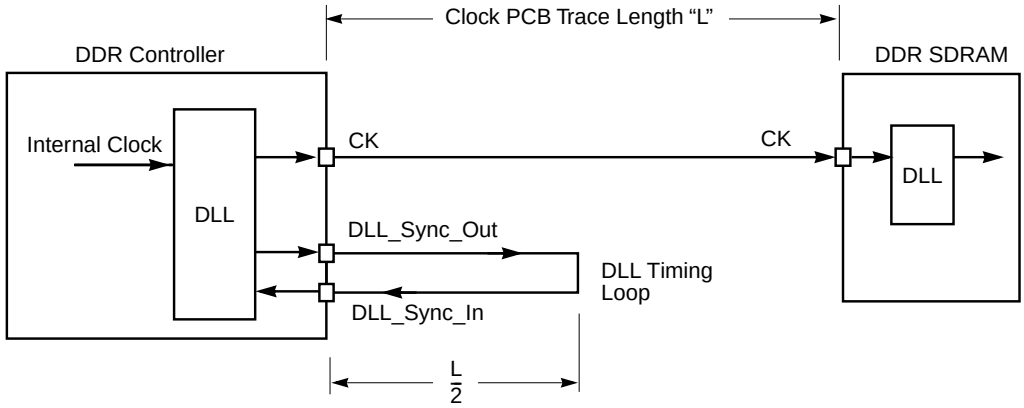


Figure 9-22. Controller DLL Timing Loop

Figure 9-23 shows an example DDR SDRAM configuration with four physical banks and Figure 9-24 shows some typical signal connections.

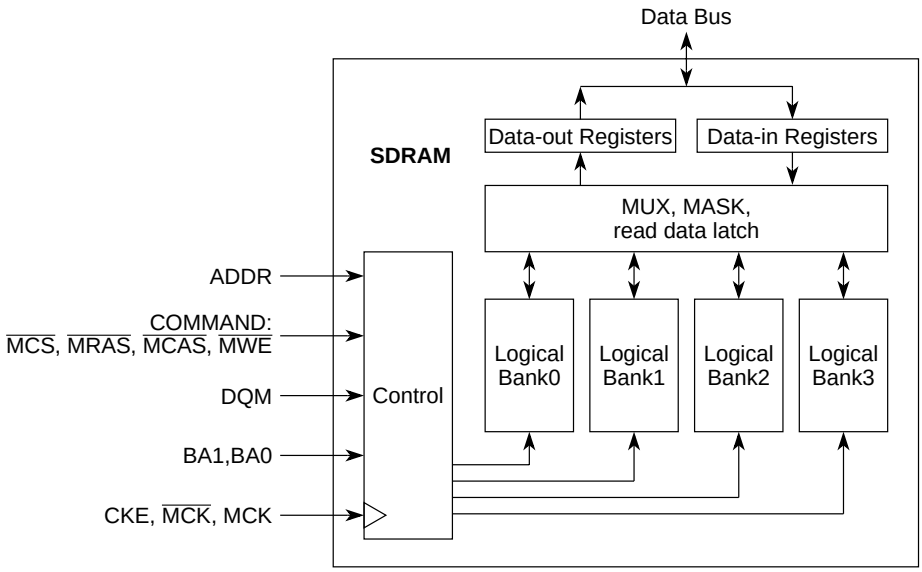


Figure 9-23. Typical Dual Data Rate SDRAM Internal Organization

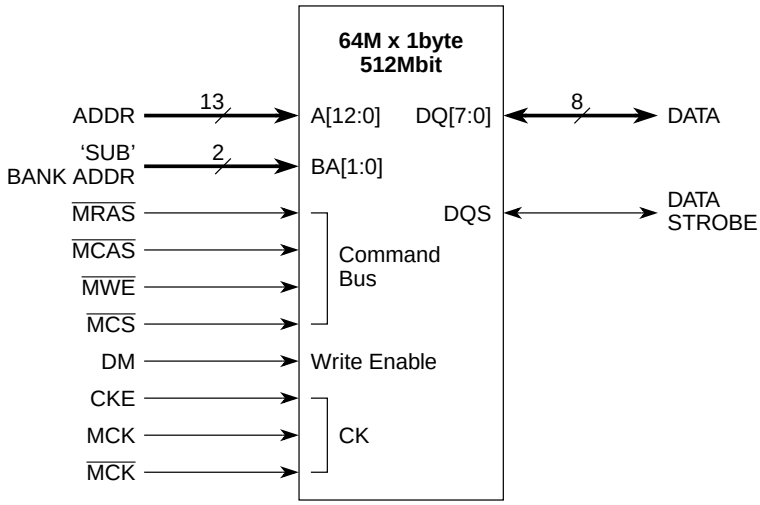
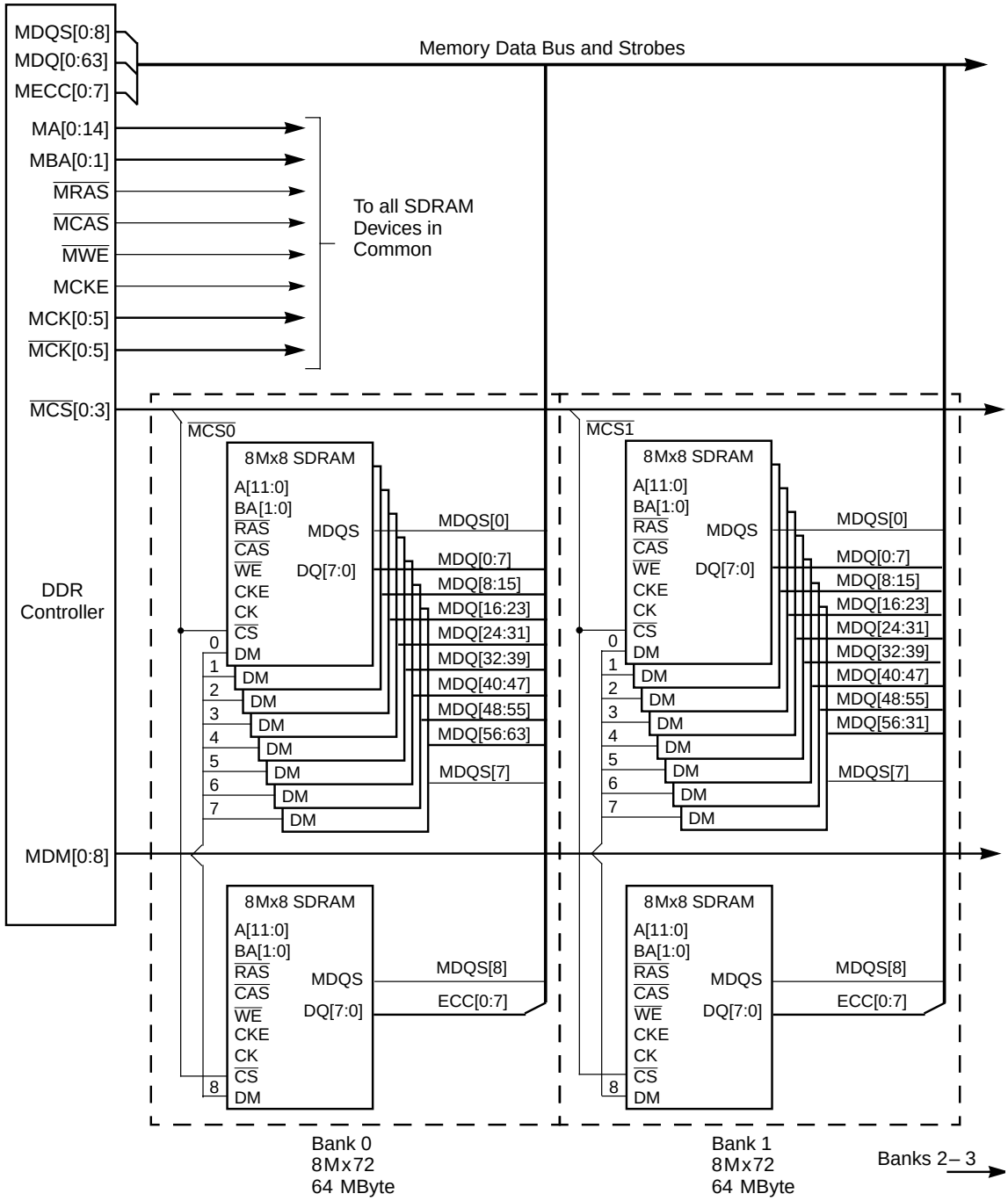


Figure 9-24. Typical DDR SDRAM Interface Signals

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 MPC8560 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 MPC8560 DDR memory controller drives 14 address pins, but in this example the DDR SDRAM devices use only 12 bits.



1. All signals are connected in common (in parallel) except for MCS[0:3],MCK[0:8], the DQM signals and the data bus signals.
2. Each of the MCS[0:3] signals correspond with a separate physical bank of memory.
3. Buffering may be needed if large memory arrays are used.
4. MCK[0:5] may be apportioned among all memory devices. Complementary bus is not shown.

Figure 9-25. 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 MPC8560 DDR memory controller supports physical bank sizes from 32 Mbytes to 1 Gbytes. The physical banks can be constructed using ×8, or ×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 $\overline{\text{MCS}}n$ signal for memory accesses according to the provided bank starting and

ending addresses. The memory banks are not required to be mapped to a contiguous address space.

9.4.2 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	$\overline{MRAS}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		1	0												
	$\overline{MCAS}$																				11	9	8	7	6	5	4	3	2	1	0	
14 x 10	$\overline{MRAS}$					13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																			1	0											
	$\overline{MCAS}$																					9	8	7	6	5	4	3	2	1	0	
13 x 11	$\overline{MRAS}$					12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																		1	0												
	$\overline{MCAS}$																				11	9	8	7	6	5	4	3	2	1	0	
13 x 10	$\overline{MRAS}$						12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																			1	0											
	$\overline{MCAS}$																					9	8	7	6	5	4	3	2	1	0	
13 x 9	$\overline{MRAS}$							12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																				1	0										
	$\overline{MCAS}$																						8	7	6	5	4	3	2	1	0	
12 x 10	$\overline{MRAS}$							11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																			1	0											
	$\overline{MCAS}$																					9	8	7	6	5	4	3	2	1	0	
12 x 9	$\overline{MRAS}$								11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																				1	0										
	$\overline{MCAS}$																						8	7	6	5	4	3	2	1	0	
12 x 8	$\overline{MRAS}$									11	10	9	8	7	6	5	4	3	2	1	0											
	MBA																					1	0									
	$\overline{MCAS}$																							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 occures.
- 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 ocure 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.

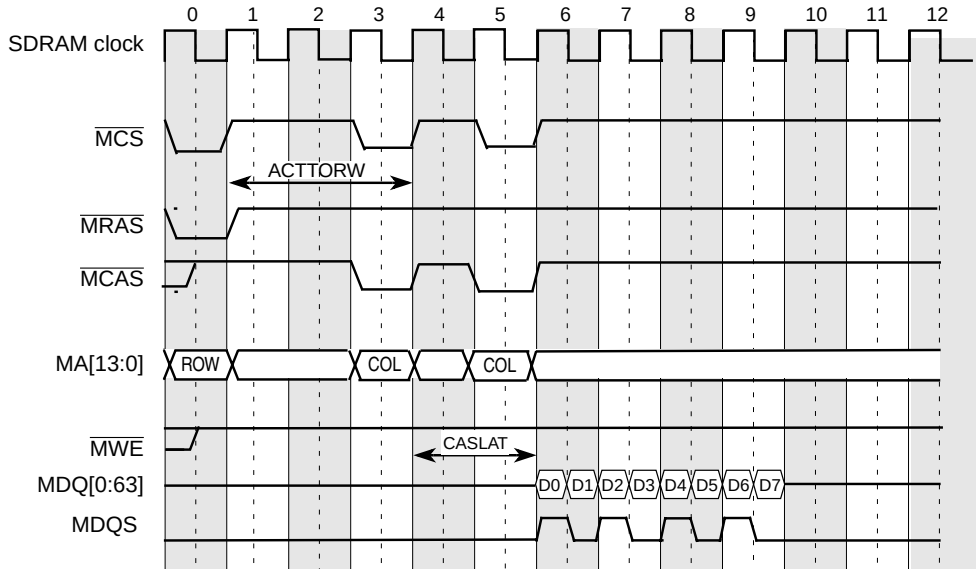


Figure 9-26. DDR SDRAM Burst Read Timing—ACTTORW = 3, $\overline{\text{MCAS}}$ Latency = 2

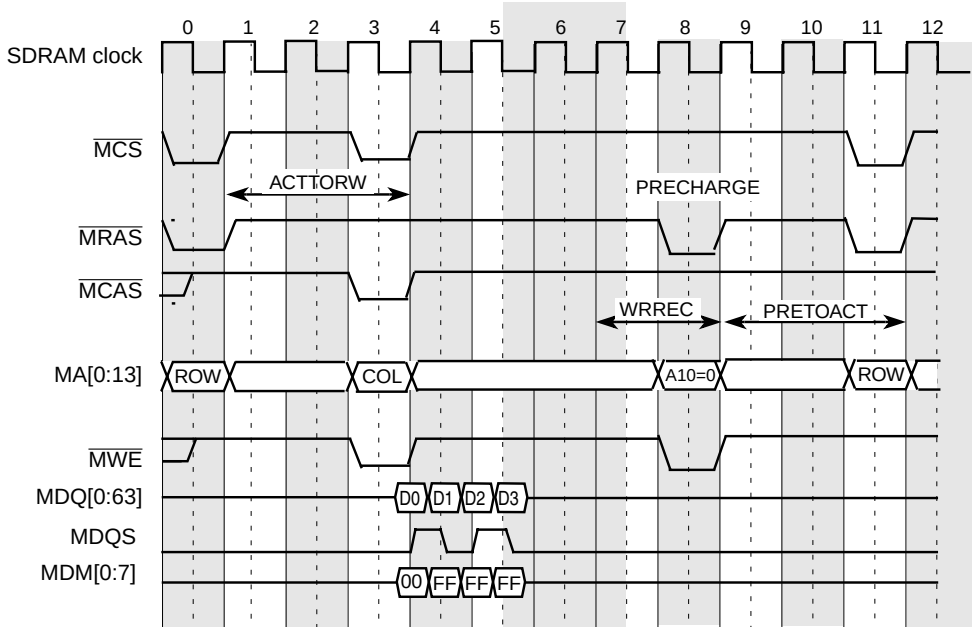


Figure 9-27. DDR SDRAM Single-Beat (Double Word) Write Timing—ACTTORW = 3

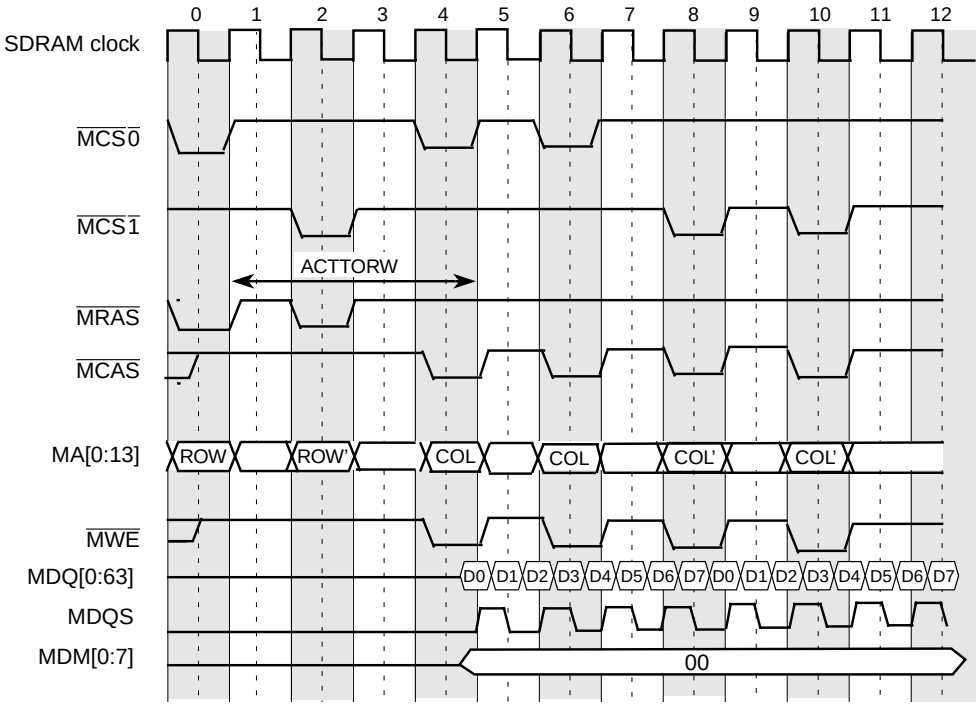


Figure 9-28. DDR SDRAM 8-Beat Burst Write Timing—ACTTORW = 4

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/$\overline{\text{MCK}}$ signal pair should drive exactly three devices.
- PCB traces for DDR clock signals should be short, all on the same layer, and of equal length and loading as the DLL timing loop.
- DDR SDRAM manufacturers provide detailed information on PCB layout and termination issues.

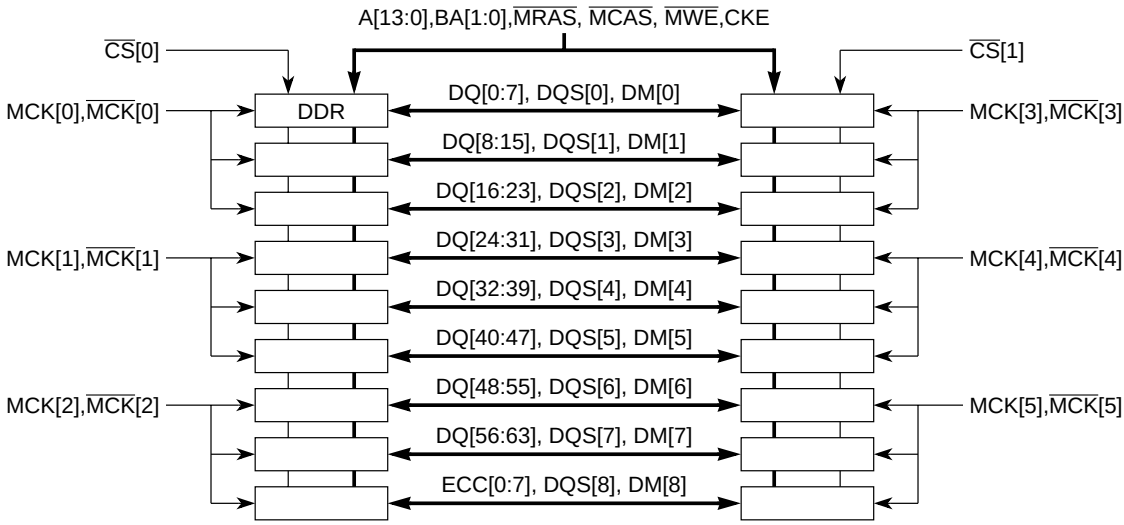


Figure 9-29. DDR SDRAM Clock Distribution Example

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.

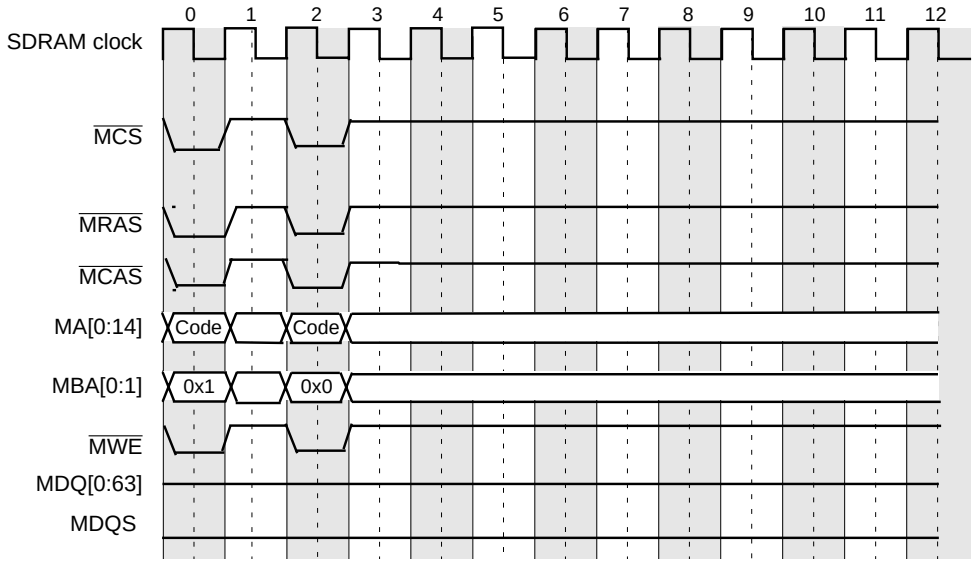


Figure 9-30. DDR SDRAM Mode-Set Command Timing

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.

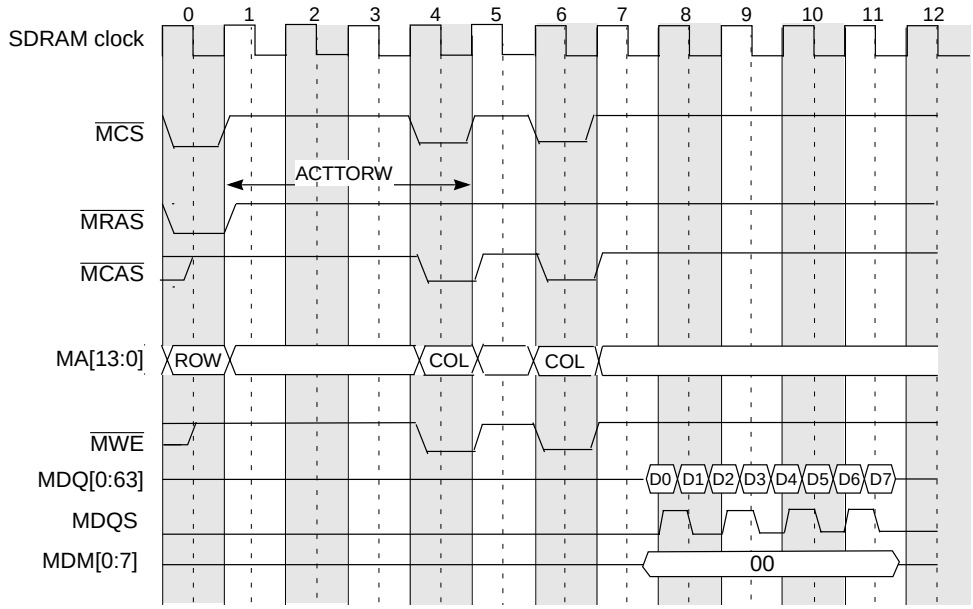


Figure 9-31. Registered DDR SDRAM DIMM Burst Write Timing

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 $1/4^{th}$ SDRAM clock periods starting with the default value of $1/4^{th}$ period delay. Figure 9-32 shows the use of the WR_DATA_DELAY parameter.

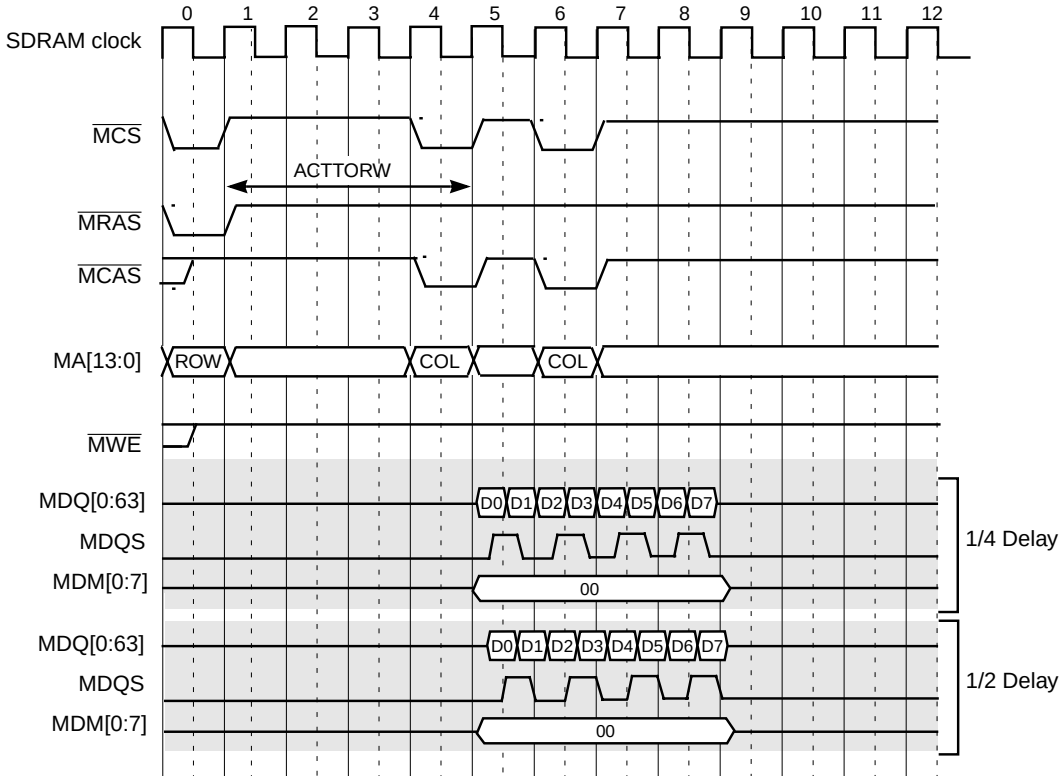


Figure 9-32. Write Timing Adjustments Example

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 simulatenously 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).

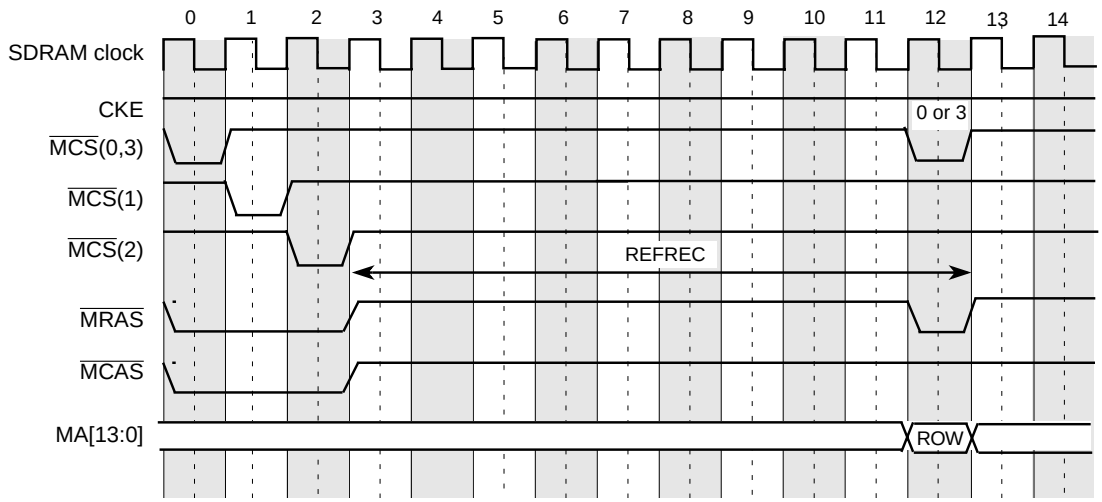


Figure 9-33. DDR SDRAM Bank Staggered Auto Refresh Timing

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.

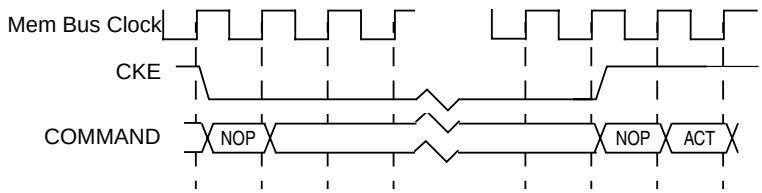


Figure 9-34. DDR SDRAM Power-Down Mode

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.

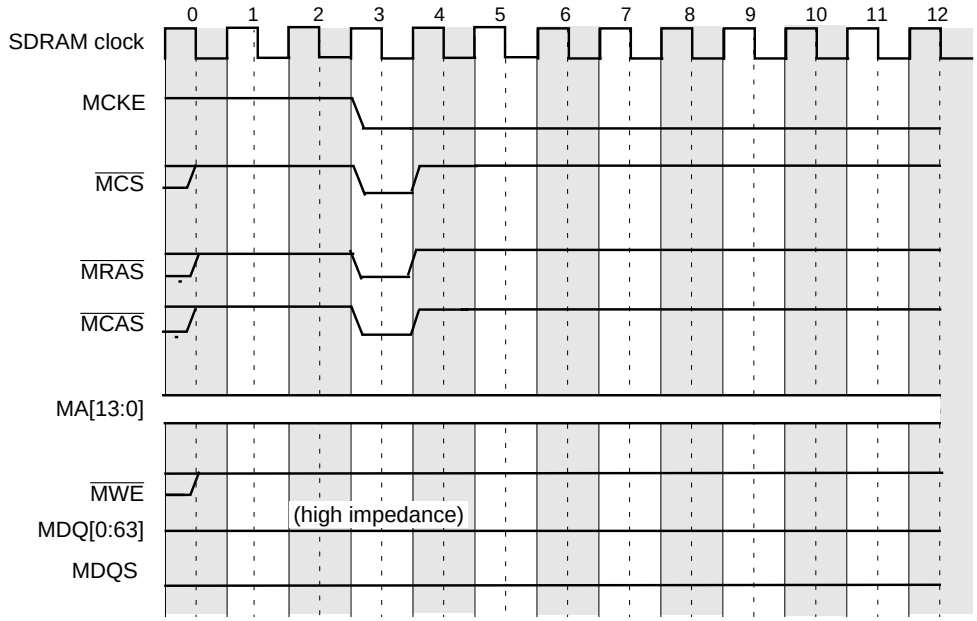


Figure 9-35. DDR SDRAM Self-Refresh Entry Timing

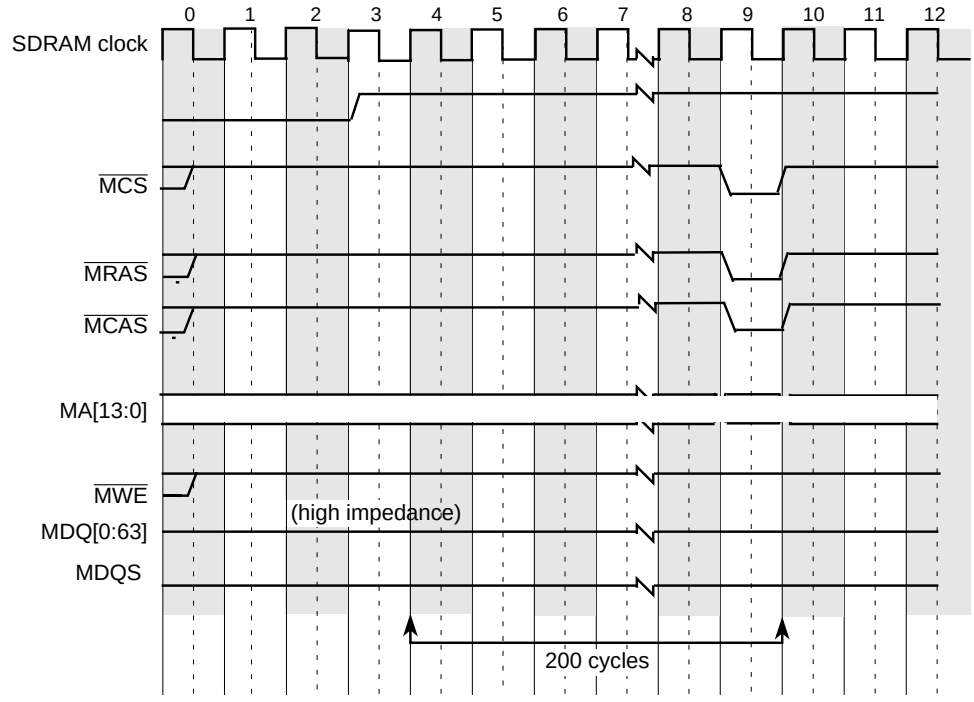


Figure 9-36. DDR SDRAM Self-Refresh Exit Timing

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 1 2 3	**0** - 1 - 2 - 3 **1** - 2 - 3 - 0 **2** - 3 - 0 - 1 **3** - 0 - 1 - 2
2 double words	0 1 2	**0 - 1** - 2 - 3 **1 - 2** - 3 - 0 **2 - 3** - 0 - 1
3 double words	0 1	**0 - 1 - 2** - 3 **1 - 2 - 3** - 0
4 double words	0 1 2 3	**0 - 1 - 2 - 3** **1 - 2 - 3 - 0** **2 - 3 - 0 - 1** **3 - 0 - 1 - 2**

[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 opens 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
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
6									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	x
4		x	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								
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 Enconding (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 Enconding (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 relevent 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 $\overline{\text{MCS}}$[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 (CSn_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 µs 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 MPC8560 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 CPM, the performance monitor, and the I^2C 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 MPC8560 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.

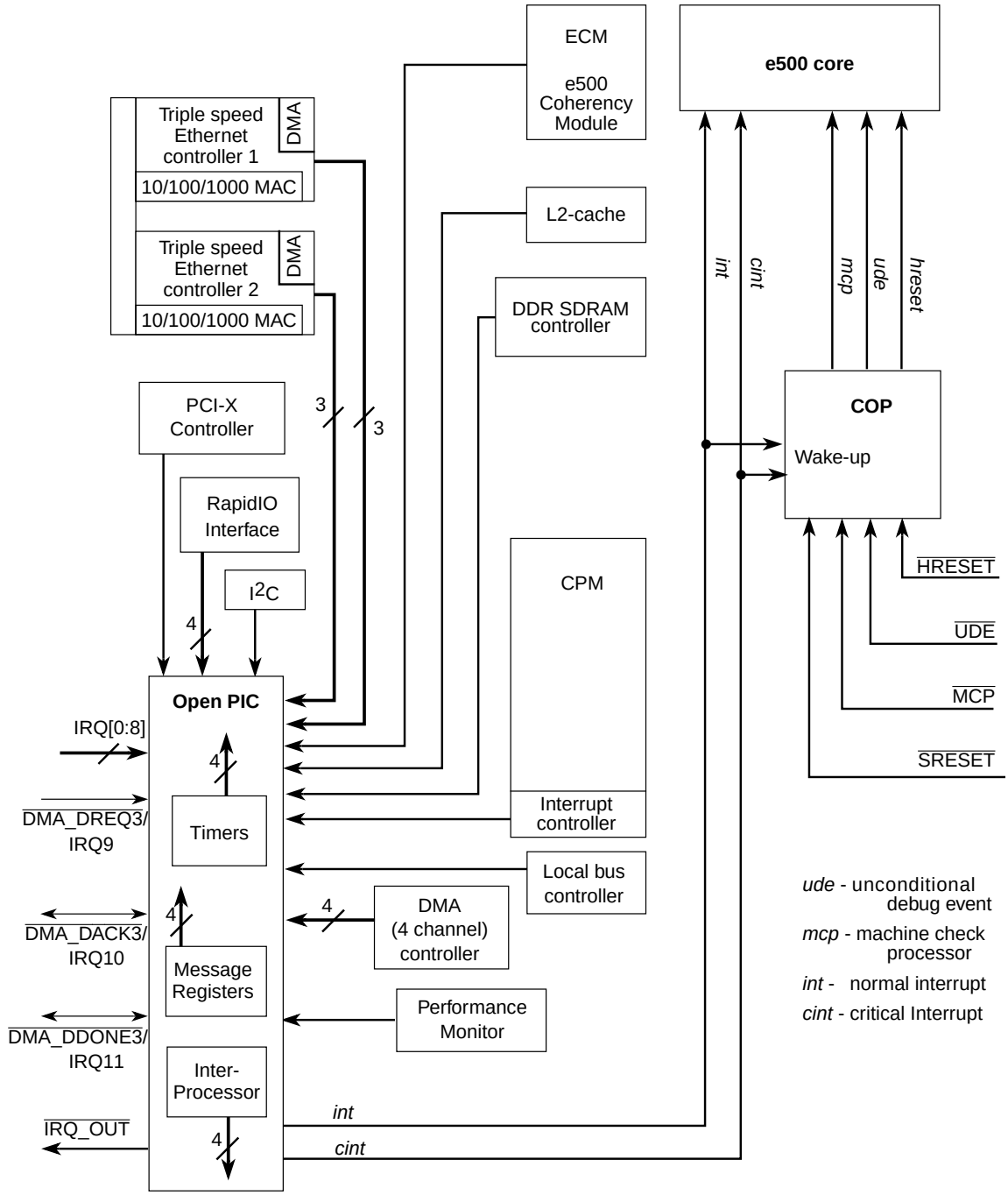


Figure 10-1. MPC8560 Interrupts Block Diagram

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 46 separate interrupts from five different sources as follows:

- 12 external—off-chip signals, IRQ[0:11]
- 22 internal—on-chip. Sources are L2, ECM, DDR, LBC, DMA,PCI/PCI-X, RapidIO, TSEC, CPM, performance monitor, and I^2C.
- 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 MPC8560.
- 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 22 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 22 internal interrupt sources for the MPC8560. 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	unused
26	unused
27	I^2C controller
28	Performance monitor interrupt
29	unused

Table 10-1. Internal Interrupt Assignments (continued)

Internal Interrupt Number	Interrupt Source
30	CPM
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 ($\overline{\text{IRQ_OUT}}$). 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/$\overline{\text{DMA_DREQ3}}$ [1]	I	External interrupt /DMA channel 3 request
IRQ10/$\overline{\text{DMA_DACK3}}$[1]	I or O	External interrupt (input)/DMA channel 3 acknowledge (output)
IRQ11/$\overline{\text{DMA_DDONE3}}$[1]	I or O	External interrupt (input)/DMA channel 3 done (output)
$\overline{\text{IRQ_OUT}}$	O	Interrupt request out
$\overline{\text{MCP}}$	I	Machine check processor
$\overline{\text{UDE}}$	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 17.3.1.10, “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 *MPC8560 Electrical Specifications*.
$\overline{\text{IRQ_OUT}}$	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 *mcp* signal that can also cause a machine check interrupt to the processor core.	
		State Meaning	Asserted—The MPC8560 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 debut event interrupt to the e500 core.	
		State Meaning	Asserted—Indicates that the MPC8560 should initiate an unconditional debut 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:

- 0x*nn*4_0000–0x*nn*4_FFF0—Global registers
- 0x*nn*5_0000–0x*nn*5_FFF0—Interrupt source configuration registers
- 0x*nn*6_0000–0x*nn*7_FFF0—Per CPU registers

Table 10-4. PIC Register Address Map

Offset	Register	Access	Reset	Section/Page
PIC Register Address Map—Global Registers				
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	$\overline{\text{IRQ_OUT}}$ summary register 0 (IRQSR0)	R	0x0000_0000	10.3.3.1/10-26
0x4_1320	$\overline{\text{IRQ_OUT}}$ 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.

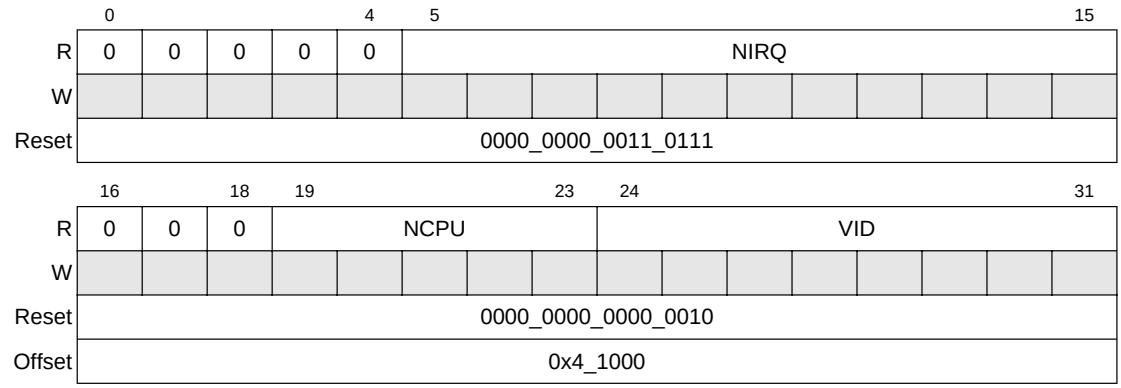


Figure 10-2. Feature Reporting Register (FRR)

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 22 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 MPC8560 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.

	0	1	2	3												15
R	RST	0	M	0	0	0	0	0	0	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	0x4_1020															

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. SeeSection 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.

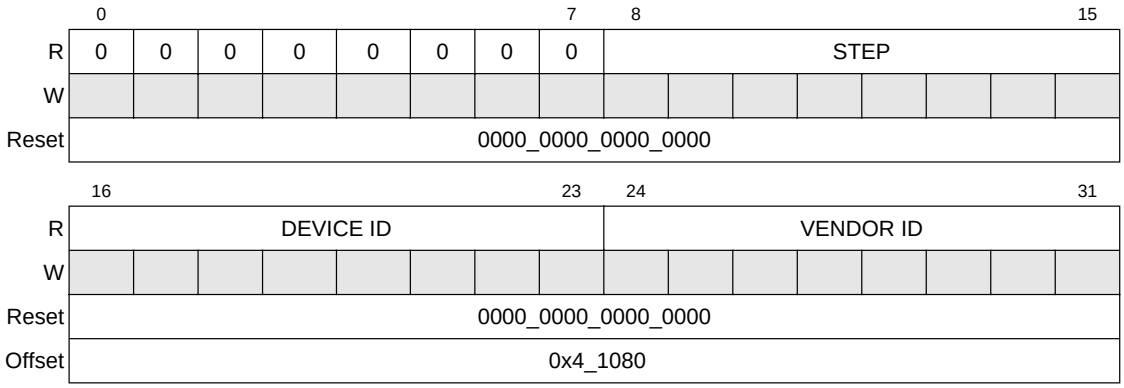


Figure 10-4. Vendor Identification Register (VIR)

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 MPC8560, only PIR[P0] is implemented.

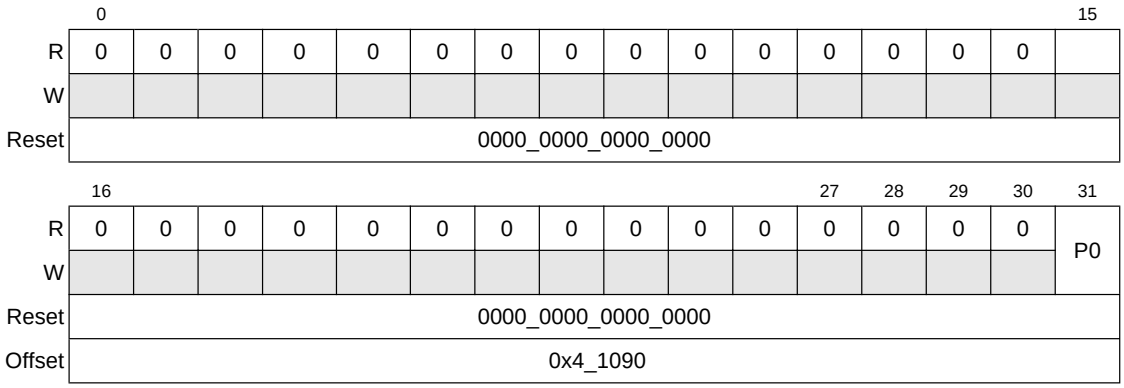


Figure 10-5. Processor Initialization Register (PIR)

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 PIVPRn[A] is set.

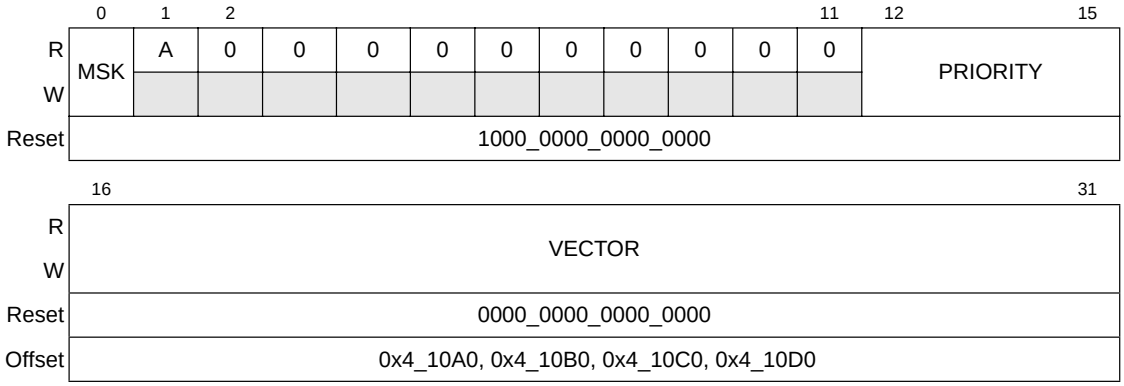


Figure 10-6. IPI Vector/Priority Register (IPIVPR*n*)

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.

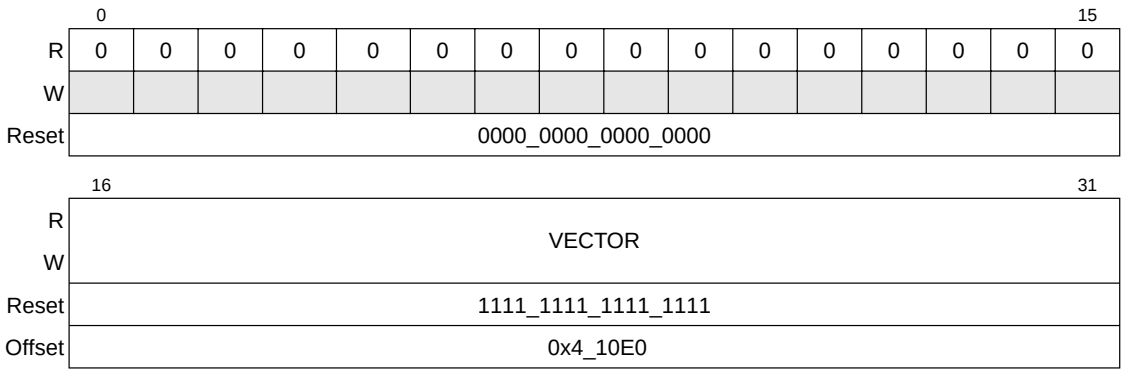


Figure 10-7. Spurious Vector Register (SVR)

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.

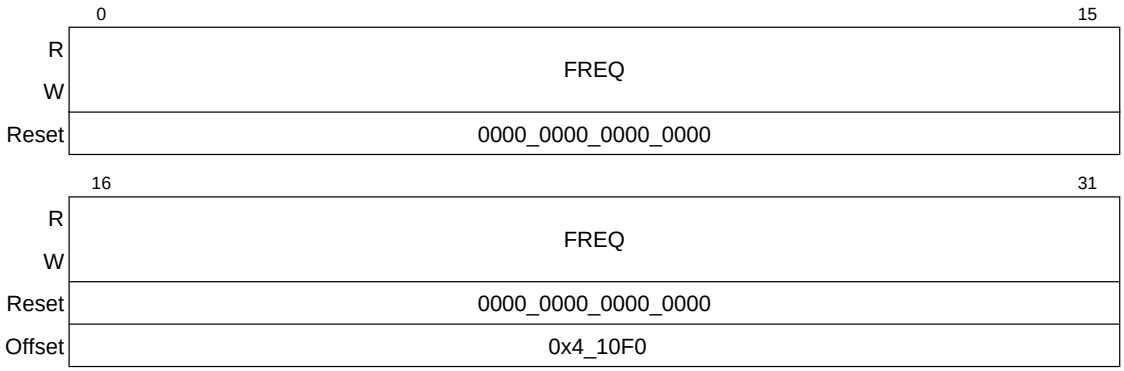


Figure 10-8. Timer Frequency Reporting Register (TFRR)

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.

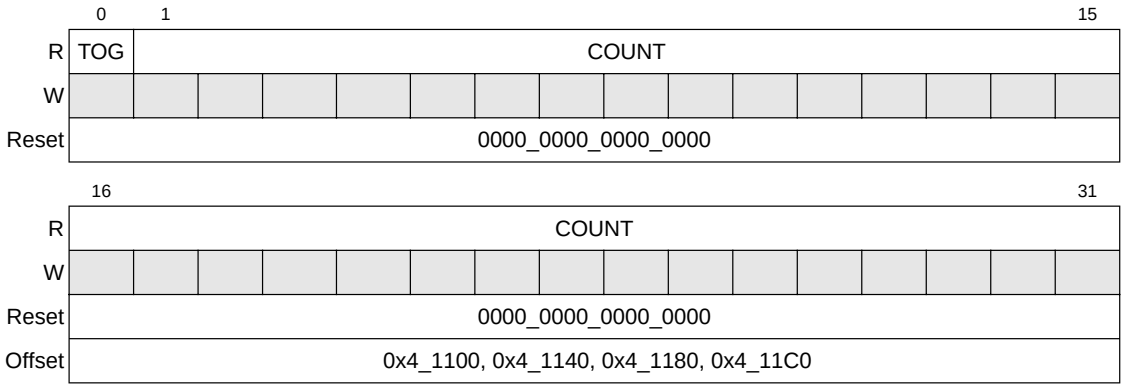


Figure 10-9. Global Timer Current Count Registers (GTCCR*n*)

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. Decremented 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.

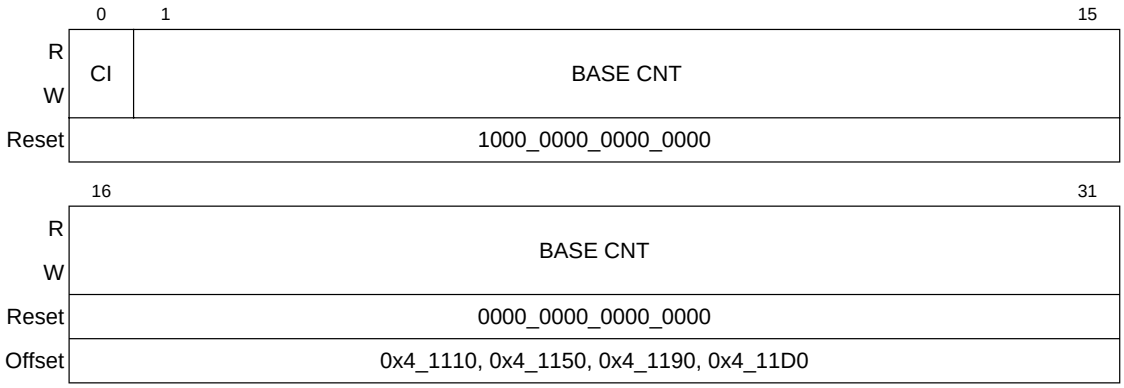


Figure 10-10. Global Timer Base Count Register (GTBCR*n*)

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.

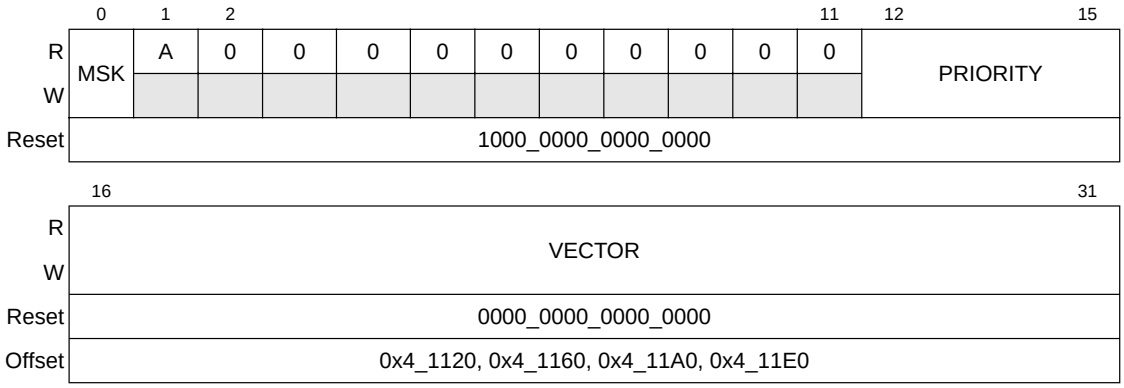


Figure 10-11. Global Timer Vector/Priority Register (GTVPR*n*)

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.

	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	0	0	0	P0
W																
Reset	0000_0000_0000_0001															
Offset	0x4_1130, 0x4_1170, 0x4_11B0, 0x4_11F0															

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 x 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

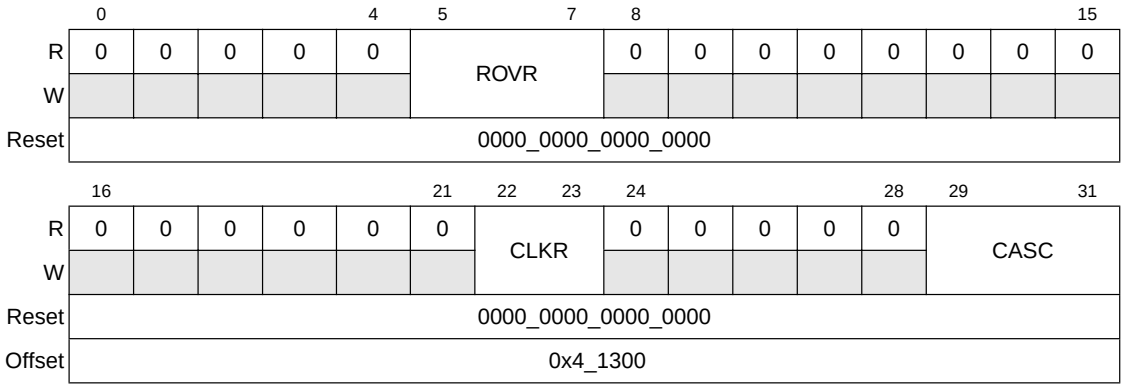


Figure 10-14. Timer Control Register (TCR)

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.

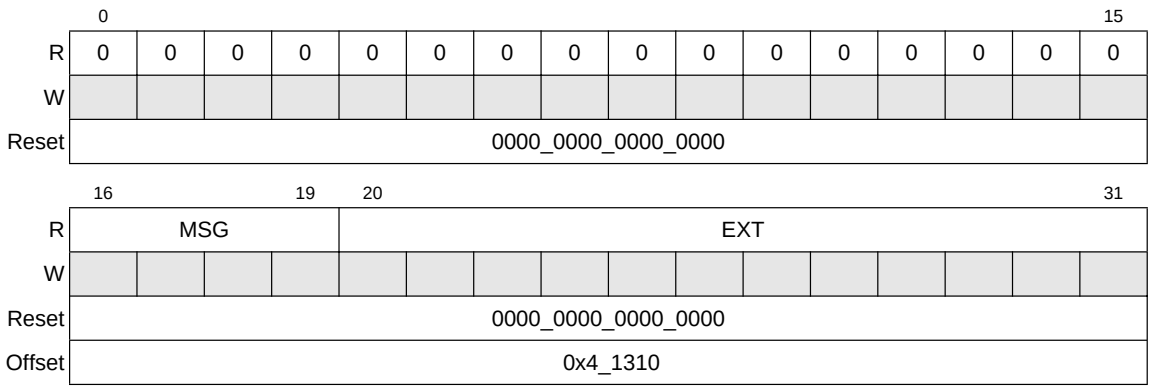


Figure 10-15. $\overline{\text{IRQ_OUT}}$ Summary Register 0

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.

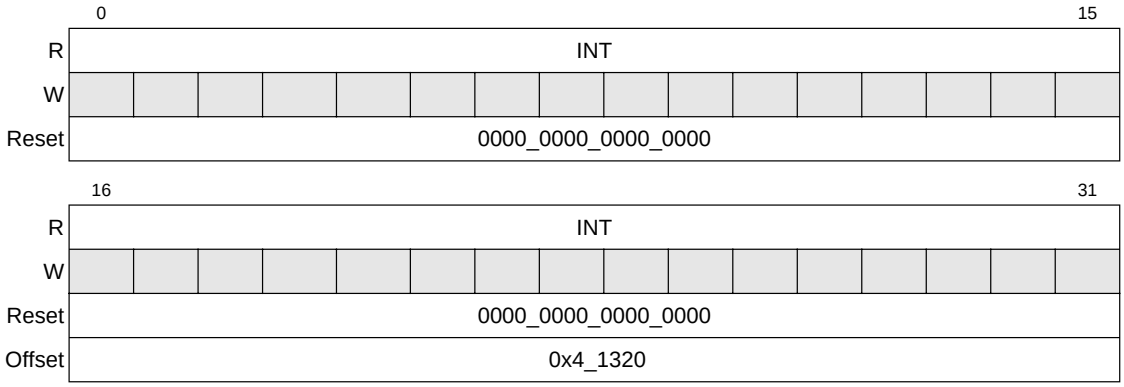


Figure 10-16. $\overline{\text{IRQ_OUT}}$ Summary Register 1

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.

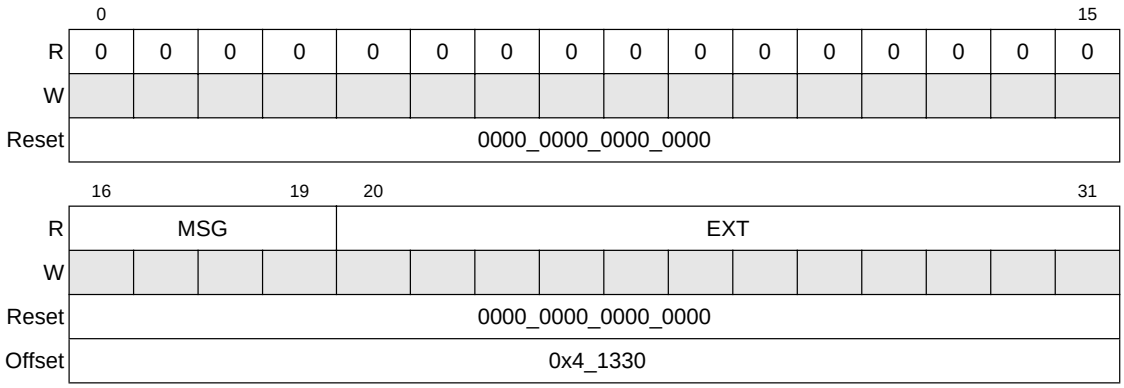


Figure 10-17. Critical Interrupt Summary Register 0

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 *cint*. 1 The corresponding interrupt is active and is directed to the *cint* (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 *cint*. 1 The corresponding interrupt is active and is directed to the *cint* (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.

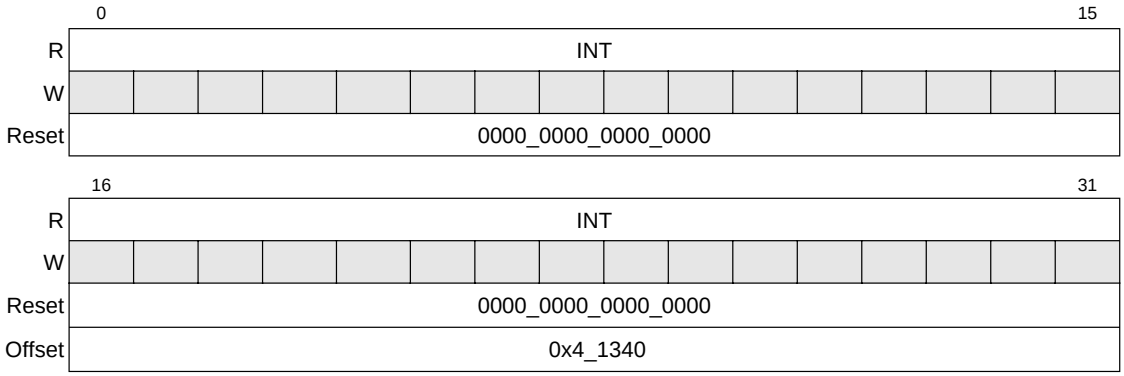


Figure 10-18. Critical Interrupt Summary Register 1

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 *cint*. 1 The corresponding interrupt is active and is directed to the *cint* (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.

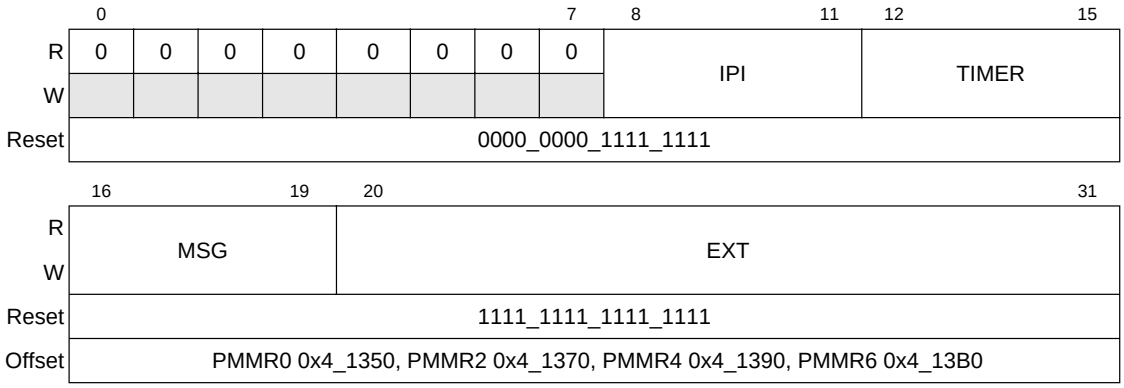


Figure 10-19. Performance Monitor Mask Registers (Even)

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.

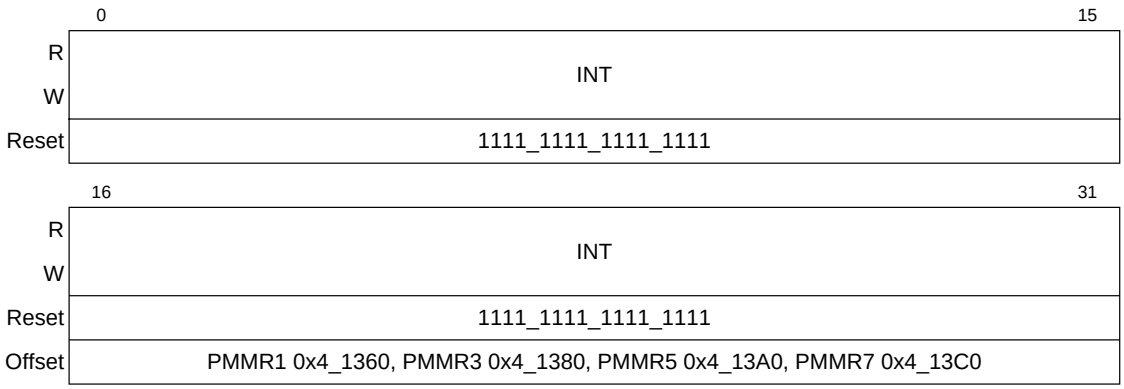


Figure 10-20. Performance Monitor Mask Registers (Odd)

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.

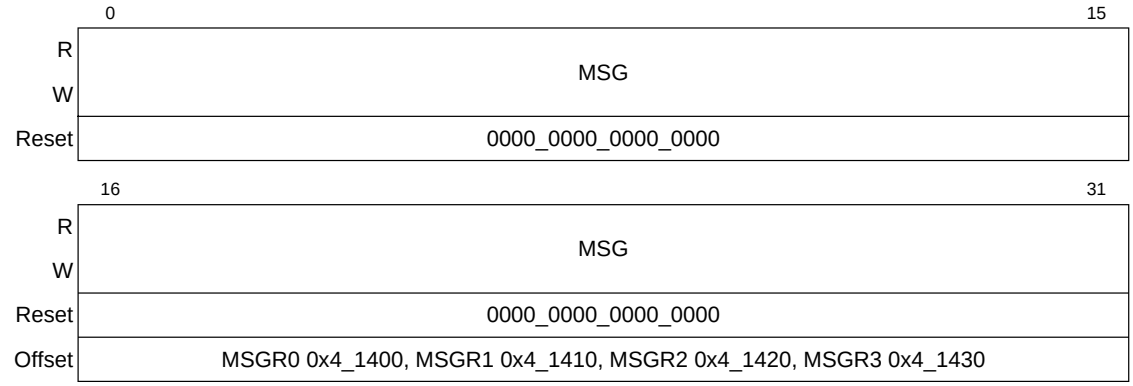


Figure 10-21. Message Registers (MSGRs)

Table 10-24 describes the MSGR fields.

Table 10-24. MSGR*n* Field Descriptions

Bits	Name	Description
0–31	MSG	Message. Contains the 32-bit message data.

10.3.5.2 Message Enable Register (MER)

The 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.

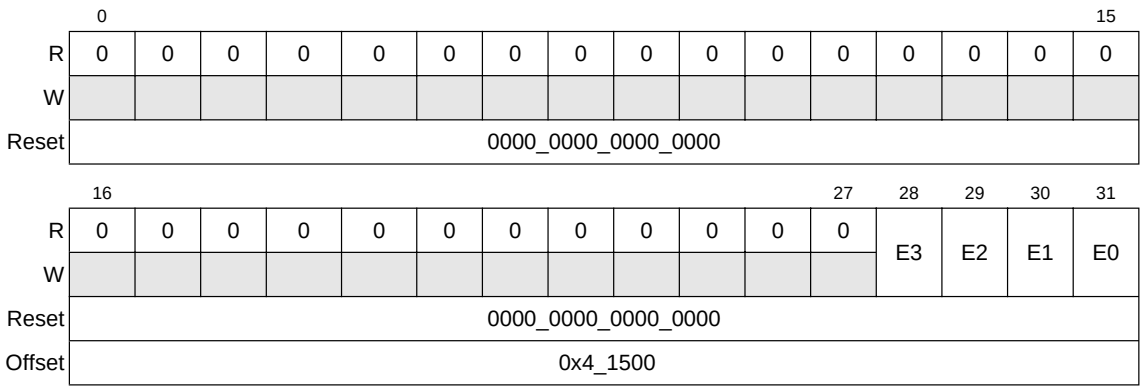


Figure 10-22. Message Enable Register (MER)

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.

	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	S3	S2	S1	S0
W																
Reset	0000_0000_0000_0000															
Offset	0x4_1510															

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.

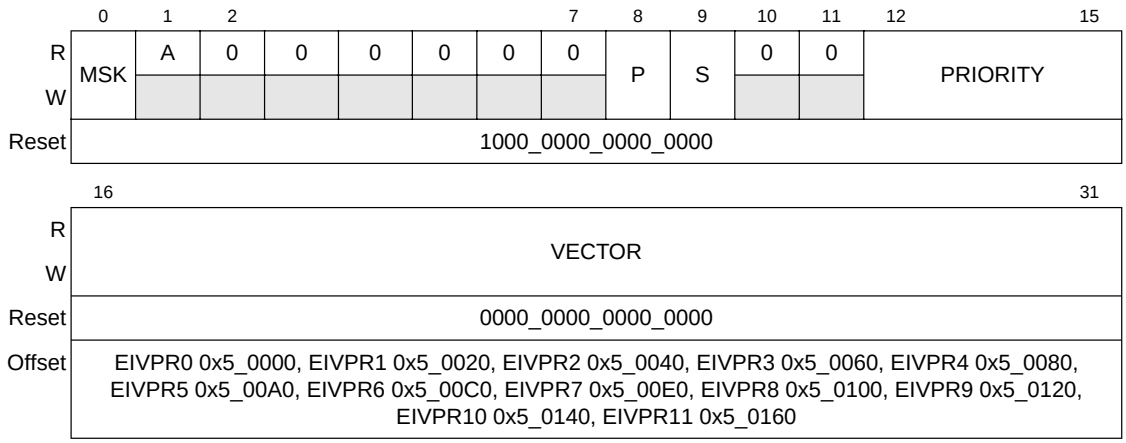


Figure 10-24. External Interrupt Vector/Priority Registers (EIVPR0 to EIVPR11)

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	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	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 *int* signal to the processor core 1 External interrupt is directed to $\overline{\text{IRQ_OUT}}$ for external service
1	CI	Critical interrupt. 0 External interrupt is serviced internally with *int* signal to the processor core 1 External interrupt is directed to the processor 0 as a critical interrupt with the *cint* 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 MPC8560 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 MPC8560 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.

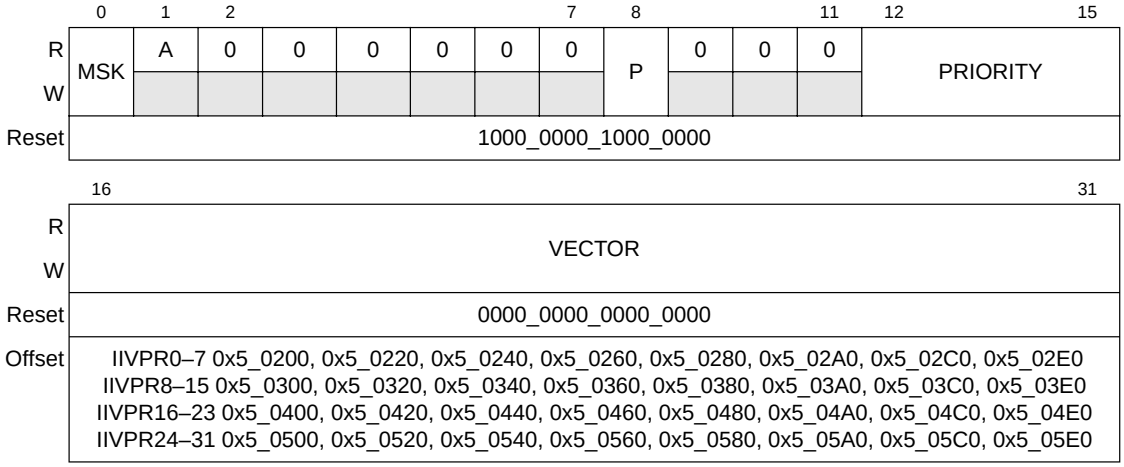


Figure 10-26. Internal Interrupt Vector/Priority Registers (IIVPRs)

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	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 *int* signal to the processor core 1 Internal interrupt is directed to $\overline{\text{IRQ_OUT}}$ 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 *int* signal to the processor core 1 Internal interrupt is directed to the processor 0 as a critical interrupt with the *cint* 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 MCP8560 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.

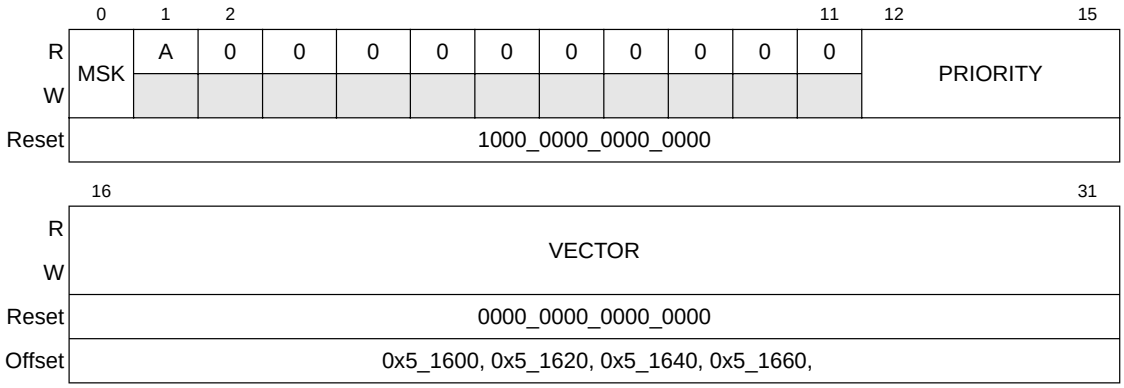


Figure 10-28. Messaging Interrupt Vector/Priority Registers (MIVPRs)

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.

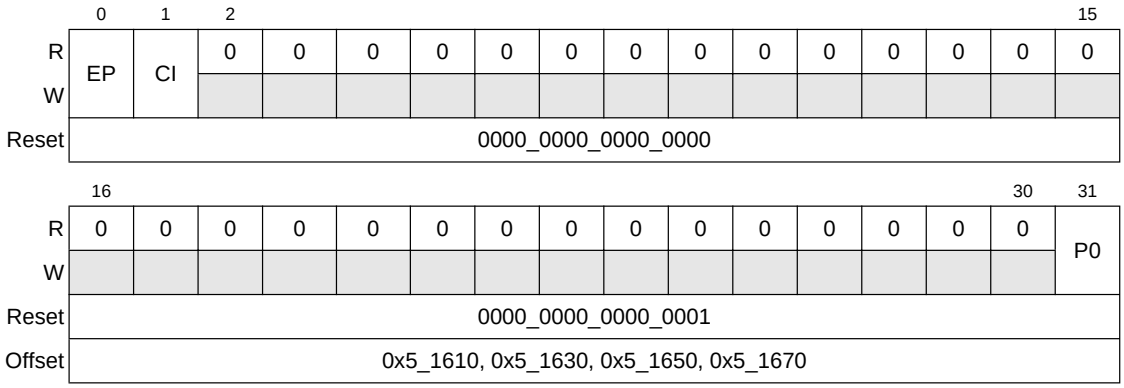


Figure 10-29. Messaging Interrupt Destination Registers (MIDRs)

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 *int* signal to the processor core. 1 Message interrupt is directed to $\overline{\text{IRQ_OUT}}$ 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 *int* signal to the processor core. 1 Message interrupt is directed to the processor 0 as a critical interrupt with the *cint* 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 MCP8560 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 MPC8560 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 MPC8560 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

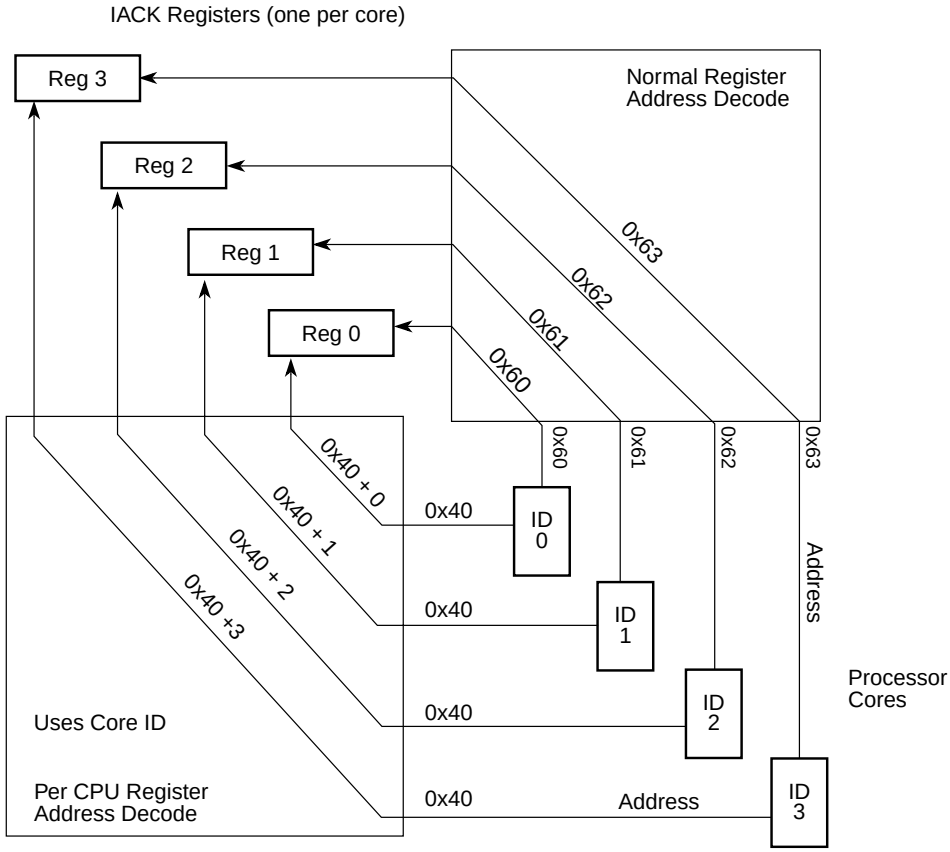


Figure 10-30. Per CPU Register Address Decoding in a Four-Core Device

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.

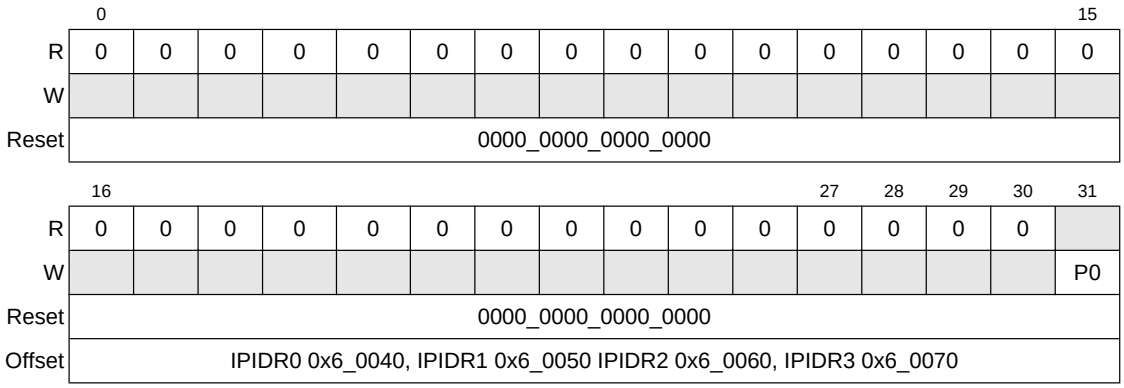


Figure 10-31. Interprocessor Interrupt Dispatch Registers (IPIDR0–IPIDR3)

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.

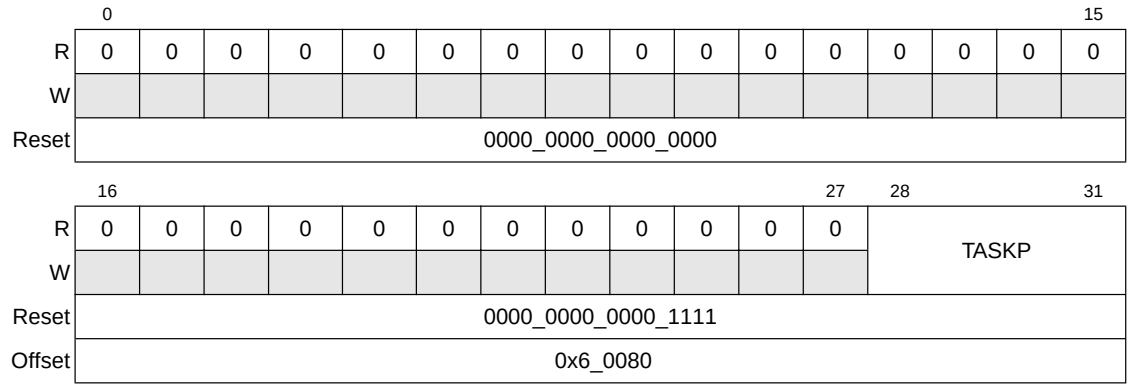


Figure 10-32. Processor Current Task Priority Register (CTPR)

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.

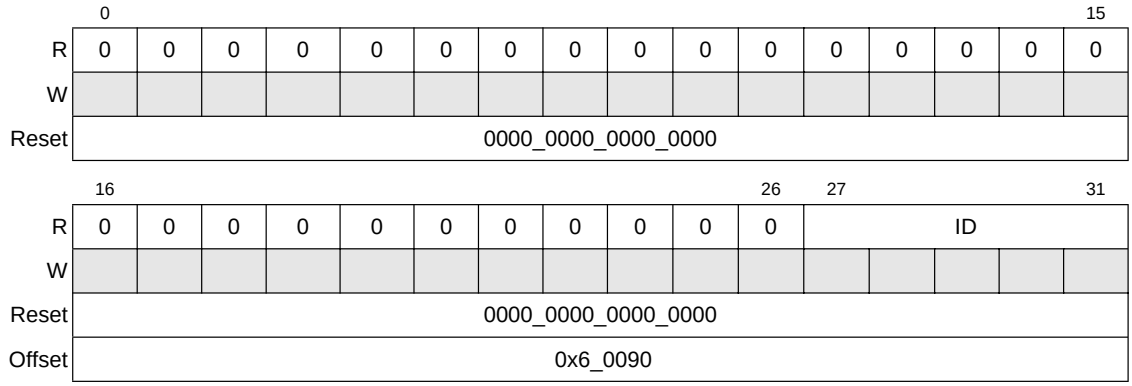


Figure 10-33. Processor Who Am I Register (WHOAMI)

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.

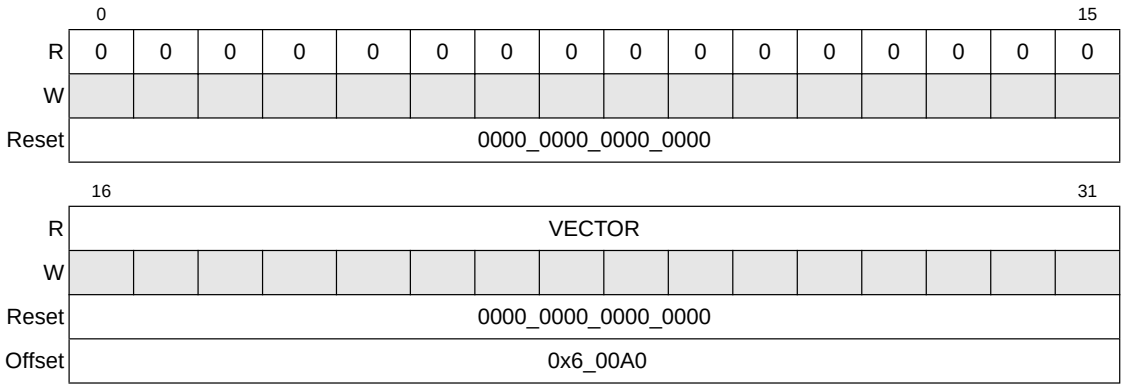


Figure 10-34. Processor Interrupt Acknowledge Register (IACK)

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.

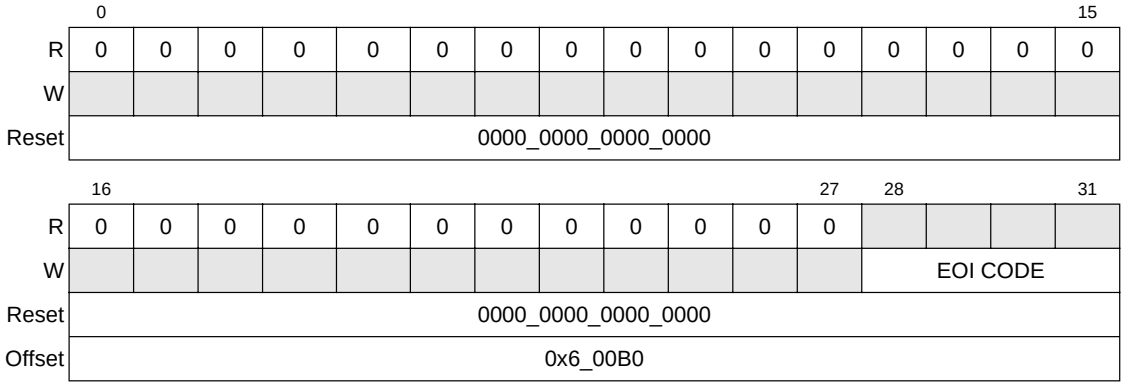


Figure 10-35. End of Interrupt Register (EOI)

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.

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.

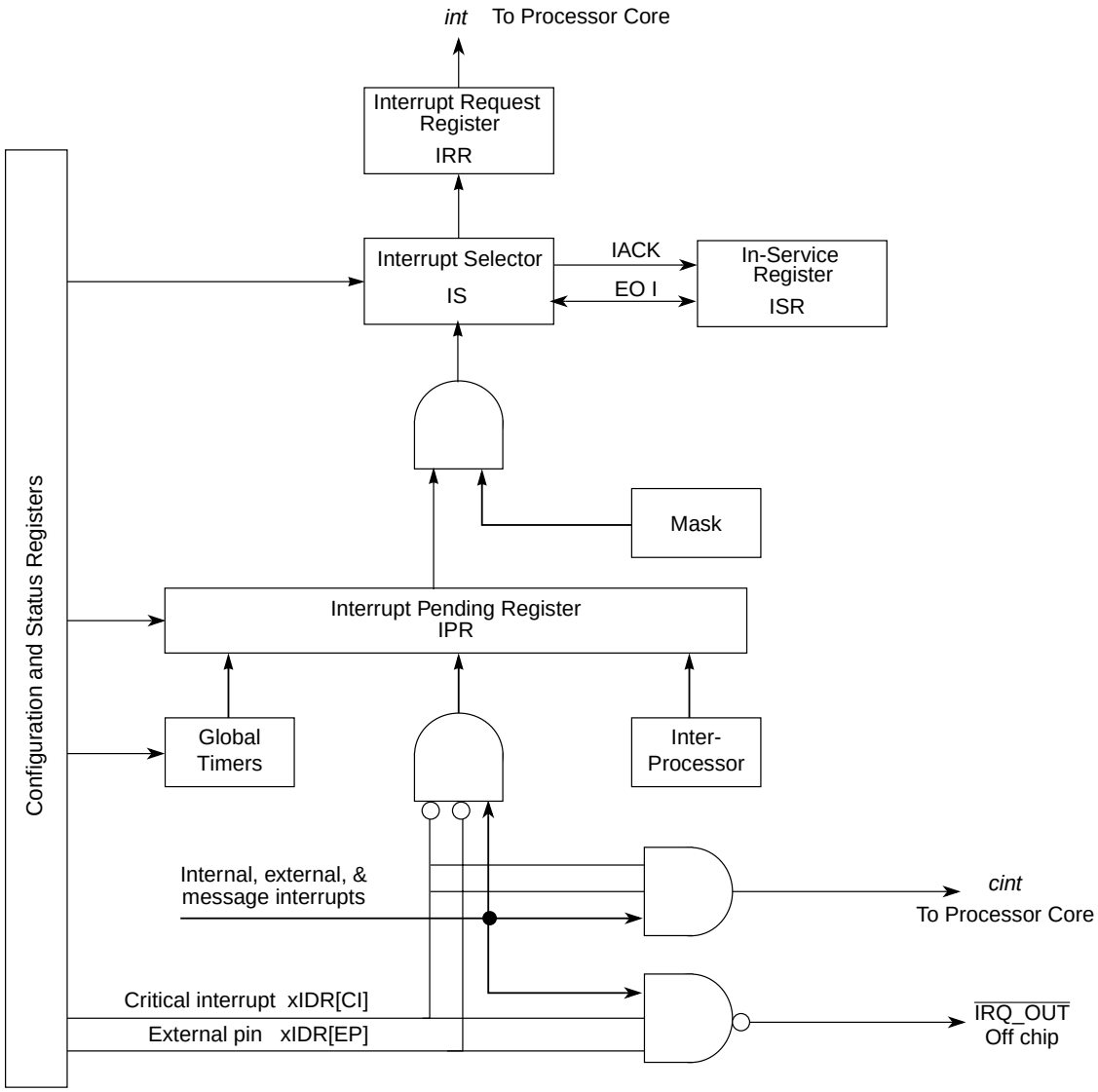


Figure 10-36. PIC Interrupt Processing Block Diagram

10.4.1 Interrupt Source Priority

Each interrupt source is assigned a priority value through its corresponding vector/priority register. Priority values are 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:

- 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
I^2C Interface

11.1 Overview

The inter-IC (IIC or I^2C) 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 I^2C 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.

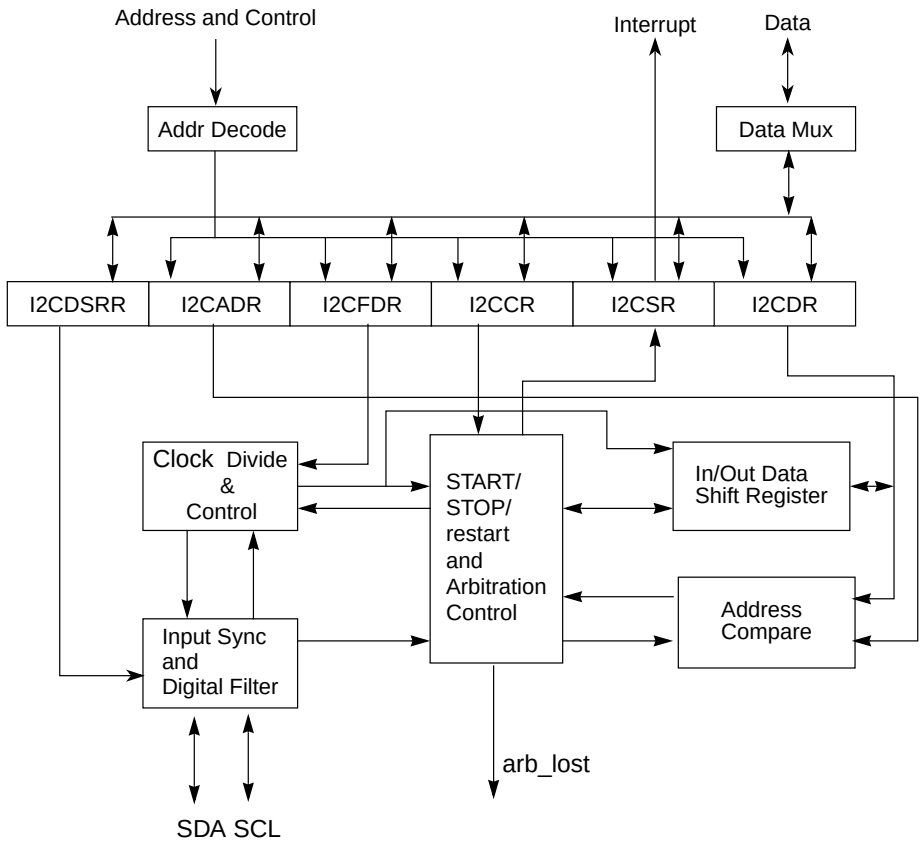


Figure 11-1. I²C Block Diagram

11.1.1 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 I^2C is the driver of the SDA line. It cannot use its own slave address as a calling address. The I^2C cannot be a master and a slave simultaneously.
- Slave mode. The I^2C is not the driver of the SDA line. The module must be enabled before a START condition from a non-I^2C 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 I^2C-specific.
- Repeated START condition. A START condition that is generated without a STOP condition to terminate the previous transfer. This mode is I^2C-specific.
- STOP condition. The master can terminate the transfer by generating a STOP condition to free the bus. This mode is I^2C-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/$\overline{W}$ 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 MPC8560 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 MPC8560 after the MPC8560 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 I^2C 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 I^2C interface.

Table 11-1. I^2C Interface Signal Description

Signal Name	Idle State	I/O	State Meaning
Serial Clock (SCL)	HIGH	I	When the I^2C 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^2C module drives SCL along with SDA when transmitting. As a slave, the I^2C module drives SCL low for data pacing.
Serial Data (SDA)	HIGH	I	When the I^2C module is idle or in a receiving mode, SDA defaults as an input. The unit receives data from other I^2C 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^2C 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 *MPC8560 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 I^2C controller.

Table 11-2. External I^2C Interface Signal—Detailed Signal Descriptions

Signal	I/O	Description	
SDA	I/O	Serial Data. This signal performs as an input when the MPC8560 is in a receiving mode. SDA also performs as an output signal when the MPC8560 is transmitting (as an I^2C 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 MPC8560 is programmed as an I^2C slave. SCL also performs as an output when the MPC8560 is programmed as an I^2C 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^2C 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 I^{2}C-specific registers and their addresses.

Table 11-3. I^{2}C 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 I^{2}C 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 I^{2}C data register (I2CDR).

11.3.1.1 I^{2}C Address Register (I2CADR)

Figure 11-2 shows the I2CADR register, which contains the address to which the I^{2}C 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 I^{2}C module is in master mode.

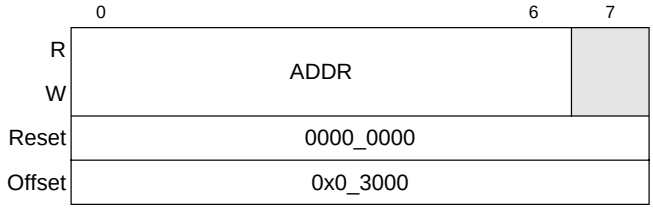


Figure 11-2. I^{2}C Address Register (I2CADR)

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 I^{2}C Frequency Divider Register (I2CFDR)

Figure 11-3 shows the sampling rate and the clock bit rate for the I^{2}C unit.

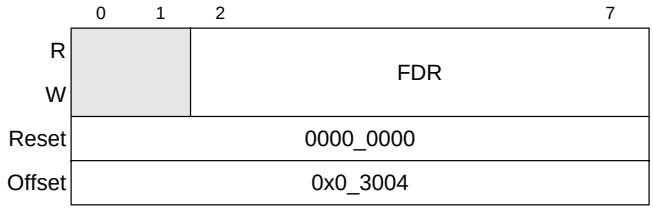


Figure 11-3. I^{2}C Frequency Divider Register (I2CFDR)

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:

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.

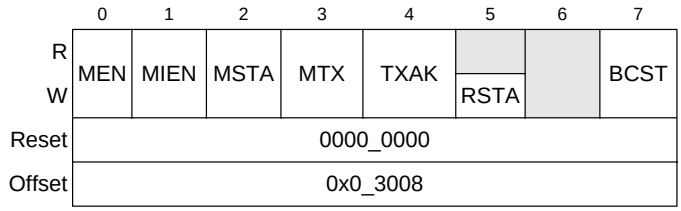


Figure 11-4. I²C Control Register (I2CCR)

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 I²C module. 0 The module is reset and disabled. When low, the interface is held in reset but the registers can still be accessed. 1 The I²C module is enabled. This bit must be set before any other control register bits have any effect. All I²C registers for slave receive or master START can be initialized before setting this bit.
1	MIEN	Module interrupt enable 0 Interrupts from the I²C module are disabled. This does not clear any pending interrupt conditions. 1 Interrupts from the I²C module are enabled. An interrupt occurs provided I2CSR[MIF] is also set.
2	MSTA	Master/slave mode START 0 When this bit is changed from 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 I²C module is configured as a receiver, not a transmitter. It also does not apply to address cycles; when the device is addressed as a slave, an acknowledge is always sent. 0 An acknowledge signal (low value on SDA) is sent out to the bus at the 9th clock 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 I²C 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.

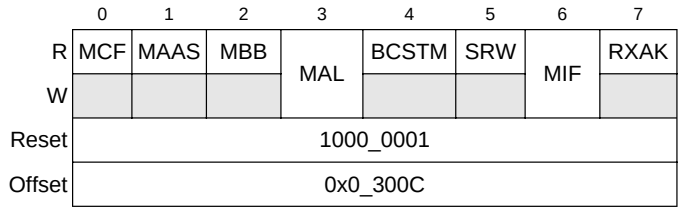


Figure 11-5. I²C Status Register (I2CSR)

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[MIEN] is set. Next, the processor must check the SRW bit and set I2CCR[MTX] accordingly. Writing to the I2CCR automatically clears this bit. 0 Not addressed as a slave 1 Addressed as a slave
2	MBB	Bus busy. Indicates the status of the bus. When a START condition is detected, MBB is set. If a STOP condition is detected, it is cleared. 0 I²C bus is idle 1 I²C bus is busy
3	MAL	Arbitration lost. Automatically set when the arbitration procedure is lost. Note that the device does not automatically retry a failed transfer attempt. 0 Arbitration is not lost. Can only be cleared by software. 1 Arbitration is lost
4	BCSTM	Broadcast match 0 There has not been a broadcast match. 1 The calling address matches with the broadcast address instead of the programmed slave address. This will also be set if the I²C 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 I^2C 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 I2CADR matches with the calling address in slave-receive mode • Arbitration is lost
7	RXAK	Received acknowledge. The value of SDA during the reception of acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates that an acknowledge signal has been received after the completion of eight bits of data transmission on the bus. If RXAK is high, it means no acknowledge signal has been detected at the 9th clock. 0 Acknowledge received 1 No acknowledge received

11.3.1.5 I^2C Data Register (I2CDR)

The I2C data register is shown in Figure 11-6.

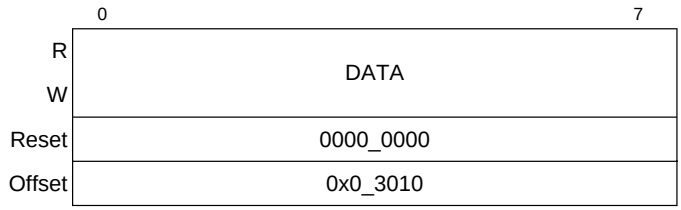


Figure 11-6. I^2C Data Register (I2CDR)

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^2C 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^2C module to receive the next byte of data on the I^2C 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.

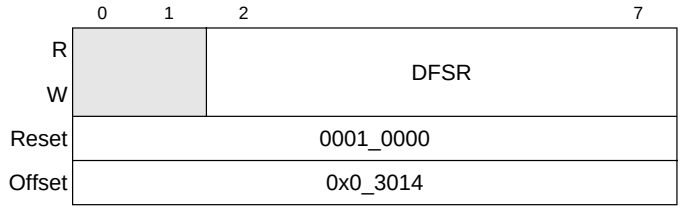


Figure 11-7. I2CDFSRR Format

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^2C 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^2C bus sample points default to the reset divisor 0x10.

11.4 Functional Description

The reset state of the I^2C is the boot sequencer mode. After the boot sequencer has completed, the I^2C interface will perform as a slave receiver. The I^2C unit always performs as a slave receiver as a default, unless explicitly programmed to be a master or slave transmitter address.

A standard I^2C 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 I^2C protocol. The details of the protocol are described in the following subsections.

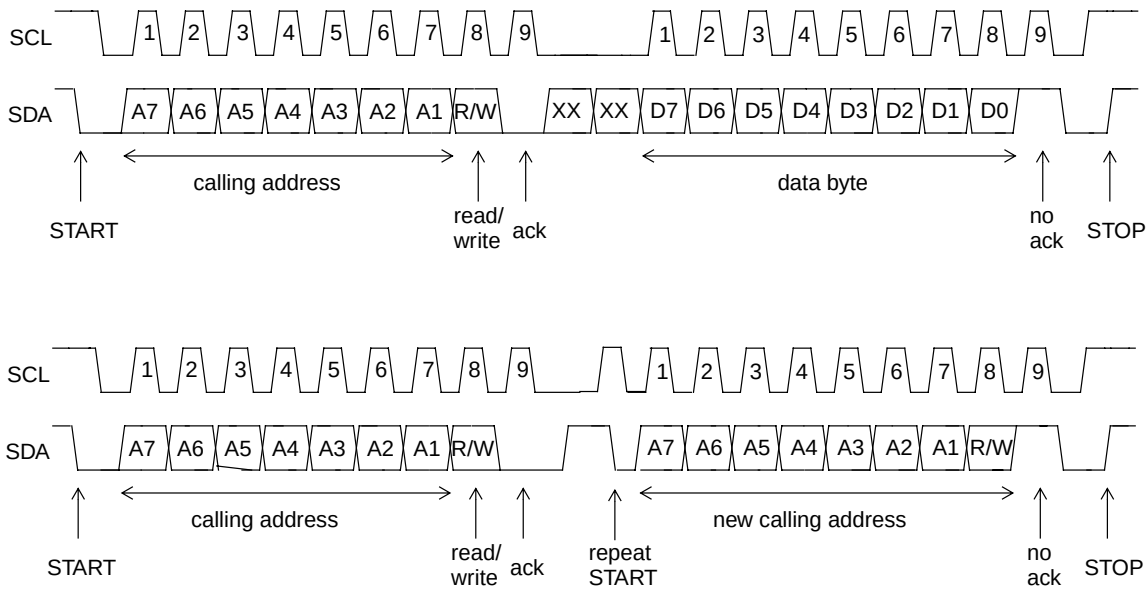


Figure 11-8. I²C Interface Transaction Protocol

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 R/$\overline{W}$ bit sent by the calling master.

The I^2C module responds to a general call (broadcast) command when I2CCR[BCST] is set. A broadcast address is always zero, however the I^2C module will not check the 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 I^2C interface is a true multiple master bus that allows more than one master device to be connected on it. If two or more masters simultaneously try to control the bus, each master’s (including the I^2C 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 I^2C unit sets the I2CSR[MAL] status bit to indicate the loss of arbitration and, as a slave, services the transaction if it is directed to itself.

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 I^2C module does not automatically retry a failed transfer attempt.

If the I^2C module is enabled in the middle of an ongoing byte transfer, the interface behaves as follows:

- Slave mode—the I^2C module ignores the current transfer on the bus and starts operating whenever a subsequent START condition is detected.
- Master mode—the I^2C module cannot tell whether the bus is busy; therefore, if a START condition is initiated, the current bus cycle can be corrupted. This ultimately results in the current bus master of the I^2C interface losing arbitration, after which bus operations return to normal.

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 I^2C Functional Blocks

The following is a description of the I^2C 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 I^2C data transfers are monitored as follows:

- START conditions are detected when an SDA fall occurs while SCL is high.
- STOP conditions are detected when and 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 I^2C contains logic that controls the output to the serial data (SDA) and serial clock (SCL) lines of the I^2C. 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 I^2C controller defaults to the boot sequencer mode. In this mode, the MPC8560 communicates with one or more EEPROMs through the I^2C interface. This EEPROM(s) contains data for the MPC8560 configuration registers. The EEPROM(s) can be programmed to initialize one or more configuration registers.

This is an extension of the standard I^2C interface that uses more address bits to allow for EEPROMs that have more than 256 bytes. During this initialization, the extended I^2C 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 I^2C interface is used, the I^2C 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 I^2C 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 MPC8560. There should be no other I^2C traffic when the boot sequencer is active.

Note that although the I^2C block defaults to the boot sequencer mode at POR, the actual boot sequencer used by the MPC8560 at power-up is determined by the LGPL3 and LGPL5 reset configuration pins of the MPC8560. Also note that the default value for the LGPL3 and LGPL5 reset configuration pins is 0b11, which corresponds to the I^2C 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 MPC8560. 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 I^2C 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 I^2C 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 I^2C asserts an error signal that can be turned into a hard reset request (*hreset_req*) outside of the I^2C module. If there is a preamble fail, the boot sequencer may continue to pull the I^2C pins low. until the hard reset occurs.

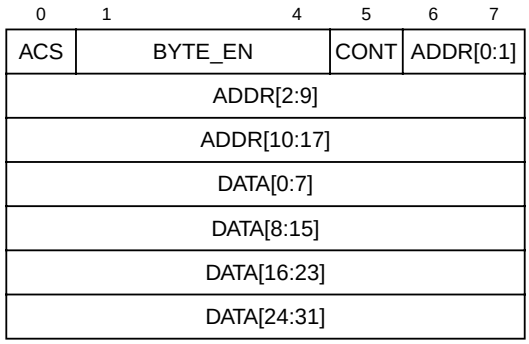


Figure 11-9. EEPROM Data Format for One Register Preload Command

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 I^2C interface. It also includes Figure 11-11, a recommended flowchart for the I^2C interrupt service routines.

The I^2C 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 I^2C register read/write access to guarantee in-order execution.

The I^2C controller does not guarantee its recovery from all illegal I^2C 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 I^2C 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 I^2C bus protocol behavior.

11.5.1 Initialization Sequence

A hard reset initializes all the I^2C registers to their default states. The following initialization sequence initializes the I^2C unit:

1. All I^2C 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 I2CADR to define the slave address for this device.
4. Modify I2CCR to select master/slave mode, transmit/receive mode, and interrupt-enable or disable.
5. Set the I2CCR[MEN] to enable the I^2C 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 I^2C 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 I^2C interrupt bit (I2CSR[MIF]) is cleared. If I2CSR[MIF] = 1 at any time, the I^2C 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 I^2C 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[MIEN] = 1). In the interrupt handler, software must do the following:

1. Clear I2CSR[MIF]
2. Read the contents of the I^2C 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 I^2C 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 I^2C 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 I^2C 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 I^2C automatically generates a STOP if I2CCR[TXAK] = 1. Therefore, I2CCR[TXAK] must be set to a 1 before allowing the I^2C module to receive the last data byte on the I^2C 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 MPC8560 to become the I^2C 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 I^2C devices to be reset. Thus, the SDA signal can be driven low by another I^2C device while the MPC8560 is coming out of reset and will stay low indefinitely. The following procedure can be used to force the MPC8560 to generate SCL so that the device driving SDA can finish its transaction:

1. Disable the I^2C and set the master bit by setting I2CCR to 0x20.
2. Enable the I^2C by setting I2CCR to 0xA0.
3. Read the I2CDR.
4. Return the MPC8560 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 I^2C interrupt service routine. Deviation from the flowchart may result in unpredictable I^2C 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 I^2C register read or write to guarantee in-order instruction execution.

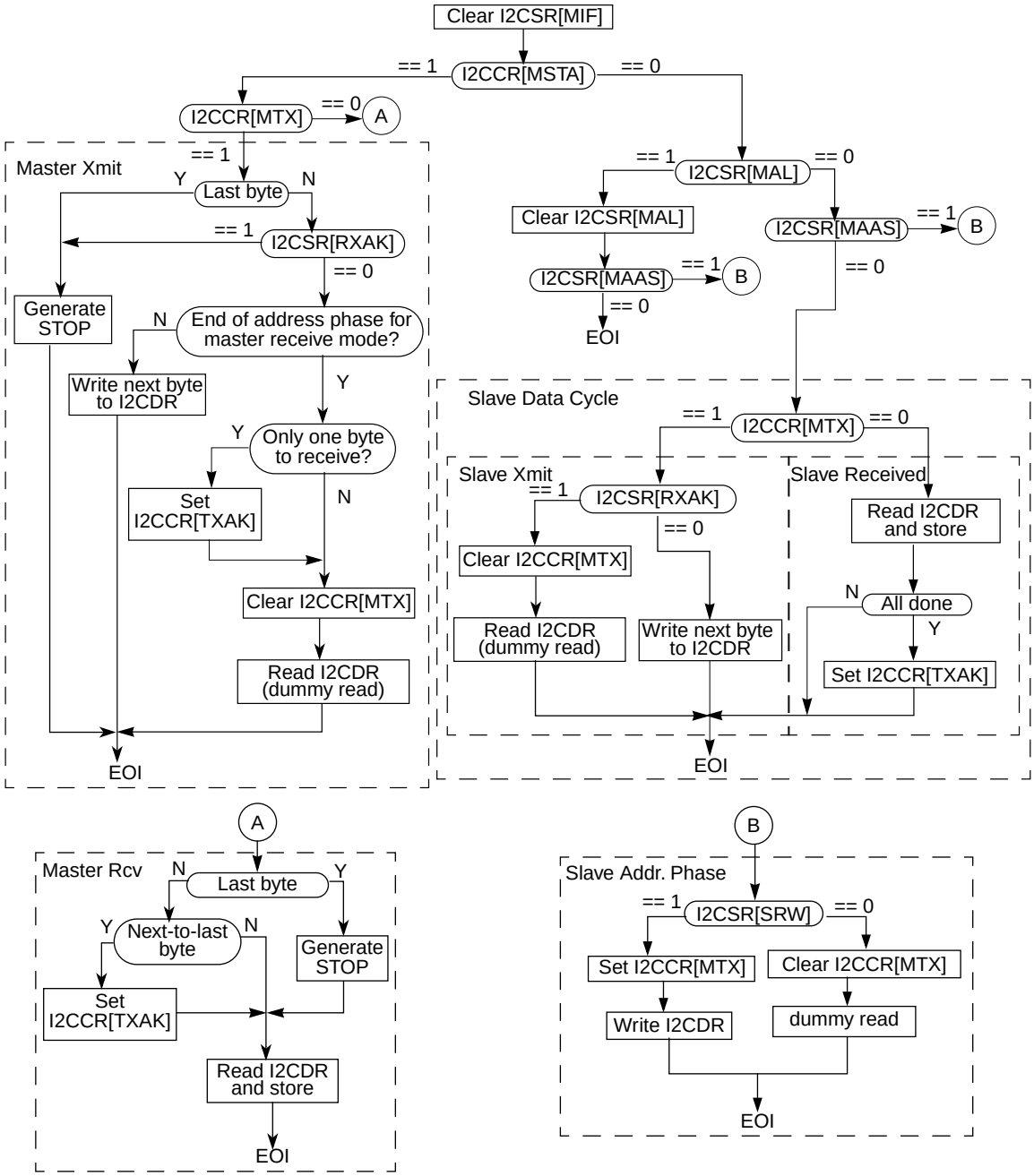


Figure 11-11. Example I²C Interrupt Service Routine Flowchart

Chapter 12
Local Bus Controller

12.1 Introduction

Figure 12-1 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.

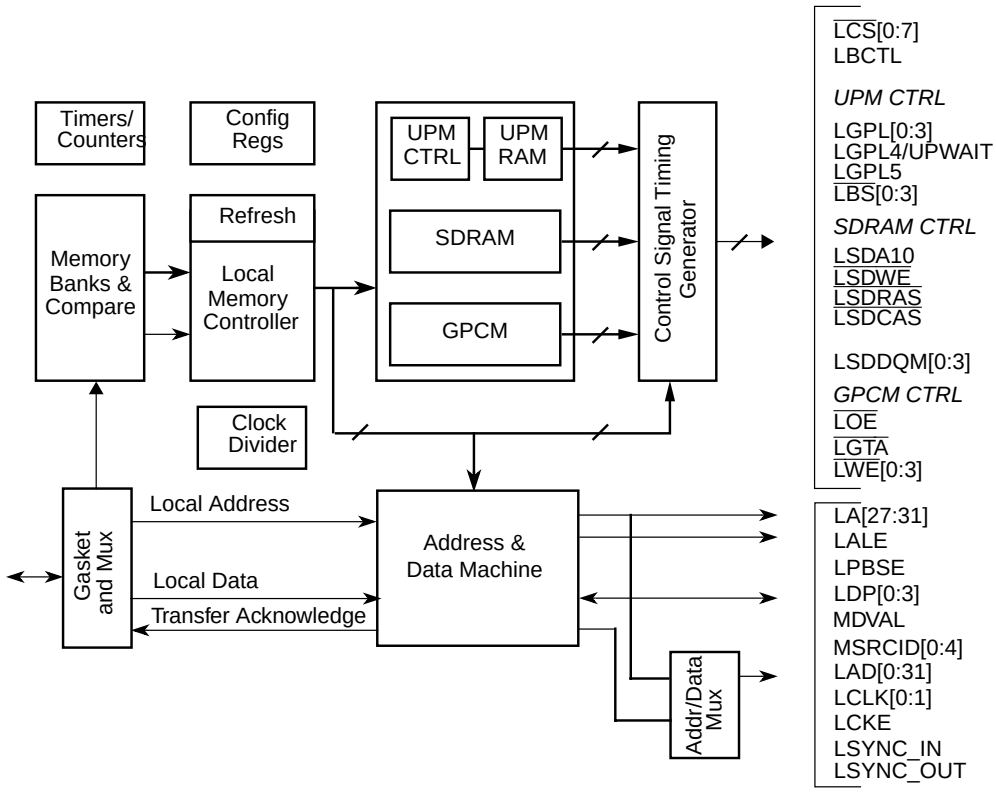


Figure 12-1. Local Bus Controller Block Diagram

12.1.1 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 (LALE) 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.

12.1.2 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, FEPROM, 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

12.1.3 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.

12.1.3.1 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.

12.1.3.2 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].

12.1.4 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.

12.1.5 References

- MPC8260 PowerQUICC II User's Manual, Chapters 4, 6, and 10, MPC8260UM/D, Rev.0, Motorola, Inc., May 1999.

12.2 External Signal Description

Table 12-1 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 12-1. 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/ $\overline{\text{LBS}}$	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
$\overline{\text{LGTA}}$/ 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 12-2 shows the detailed external signal descriptions for the local bus controller.

Table 12-2. 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—
$\overline{\text{LCS}}$[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. $\overline{\text{LCS}}$[0:7] are provided on a per-bank basis with $\overline{\text{LCS0}}$ corresponding to the chip select for memory bank 0, which has the memory type and attributes defined by BR0 and OR0.
		Timing	Assertion— Negation—
$\overline{\text{LWE}}$[0:3]/ LSDDQM[0:3]/ $\overline{\text{LBS}}$[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, $\overline{\text{LWE}}$[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. $\overline{\text{LBS}}$[0:3] are programmable byte-select signals in UPM mode. See Section 12.4.4.4, “The RAM Array,” for programming details about $\overline{\text{LBS}}$[0:3]. Negated—
		Timing	Assertion— See Section 12.4.2, “General-Purpose Chip-Select Machine (GPCM), for details regarding the timing of $\overline{\text{LWE}}$[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 12-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{LSDWE}}$/ 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—
$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$/ 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—
$\overline{\text{LSDCAS}}$/ 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—
$\overline{\text{LGTA}}$/ 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 $\overline{\text{LBS}}$[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 12-2. 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/$\overline{\text{read}}$ control for a bus transceiver connected to the LAD lines. Note that an external data buffer must not drive the LAD lines in conflict with the 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 12-2. 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 12-19 on page 12-36), 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 12-2. 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—

12.3 Memory Map/Register Definition

Table 12-3 shows the memory mapped registers of the LBC.

Table 12-3. Local Bus Controller Memory Map

Address Offset	Use	Access	Reset	Section/Page
0x0_5000	Base register 0 (BR0)	R/W	0x0000_RR01[1]	12.3.1.1/12-11
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	12.3.1.2/12-13
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	12.3.1.3/12-20
0x0_5070	UPMA mode register (MAMR)	R/W	0x0000_0000	12.3.1.4/12-20

Table 12-3. Local Bus Controller Memory Map (continued)

Address Offset	Use	Access	Reset	Section/Page
0x0_5074	UPMB mode register (MBMR)	R/W	0x0000_0000	12.3.1.4/12-20
0x0_5078	UPMC mode register (MCMR)	R/W	0x0000_0000	12.3.1.4/12-20
0x0_5084	Memory refresh timer prescaler register (MRTPR)	R/W	0x0000_0000	12.3.1.5/12-23
0x0_5088	UPM data register (MDR)	R/W	0x0000_0000	12.3.1.6/12-24
0x0_5094	SDRAM mode register (LSDMR)	R/W	0x0000_0000	12.3.1.7/12-25
0x0_50A0	UPM refresh timer (LURT)	R/W	0x0000_0000	12.3.1.8/12-27
0x0_50A4	SDRAM refresh timer (LSRT)	R/W	0x0000_0000	12.3.1.9/12-28
0x0_50B0	Transfer error status register (LTESR)	Read/ Bit-reset	0x0000_0000	12.3.1.10/12-29
0x0_50B4	Transfer error disable register (LTEDR)	R/W	0x0000_0000	12.3.1.11/12-31
0x0_50B8	Transfer error interrupt register (LTEIR)	R/W	0x0000_0000	12.3.1.12/12-32
0x0_50BC	Transfer error attributes register (LTEATR)	R/W	0x0000_0000	12.3.1.13/12-33
0x0_50C0	Transfer error address register (LTEAR)	R/W	0x0000_0000	12.3.1.14/12-34
0x0_50D0	Configuration register (LBCR)	R/W	0x0000_0000	12.3.1.15/12-34
0x0_50D4	Clock ratio register (LCRR)	R/W	0x8000_0008	12.3.1.16/12-35

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

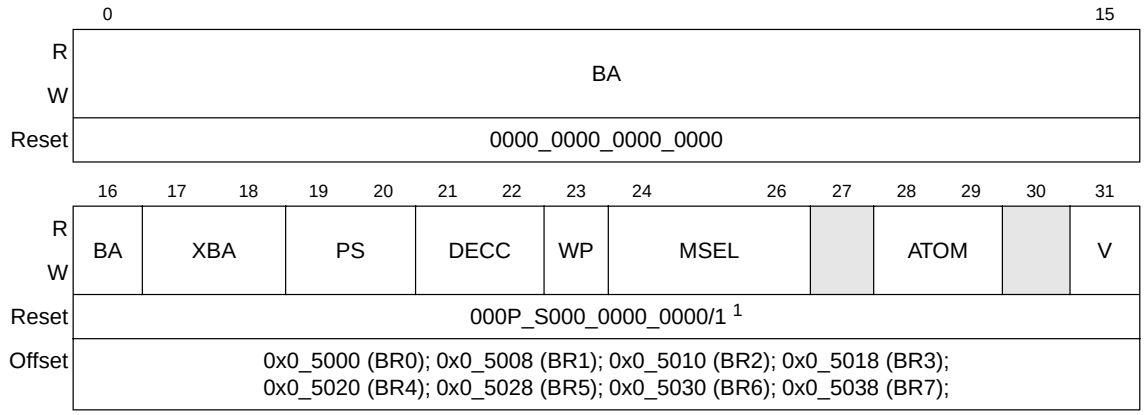
12.3.1 Register Descriptions

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 12-3 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.

12.3.1.1 Base Registers (BR0–BR7)

The base registers (BR*x*) 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 12-2 contains the field descriptions for BR*x*. 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.



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 12-2. Base Registers (BR*x*)

Table 12-4 describes BR*x* fields.

Table 12-4. BR*x* 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 12-4. BR*x* 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.

12.3.1.2 Option Registers (OR0–OR7)

The OR*x* registers define the sizes of memory banks and access attributes. The OR*x* attribute bits support the following three modes of operation as defined by BR*x*[MSEL].

- GPCM mode
- UPM mode
- SDRAM mode

The OR*x* registers are interpreted differently depending on which of the three machine types is selected for that bank.

12.3.1.2.1 Address Mask

The address mask fields of the option registers (OR*x*[XAM,AM]) mask up to 19 corresponding BR*x*[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

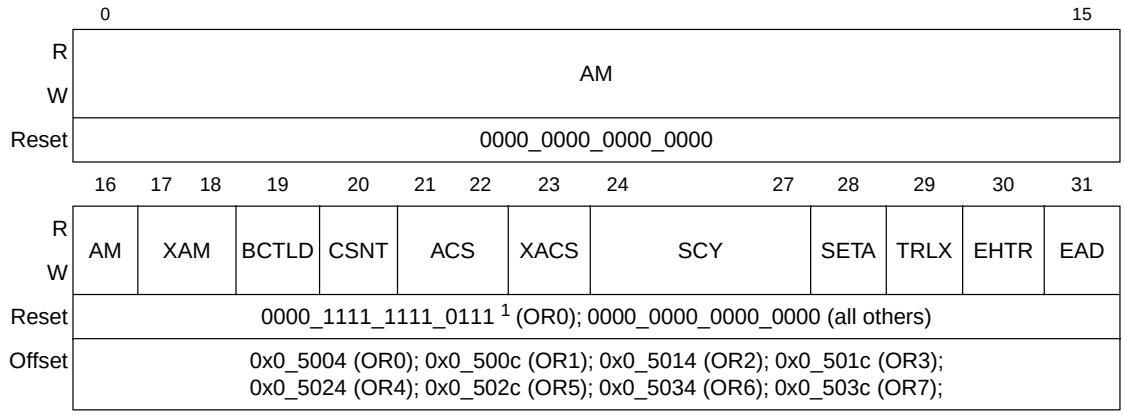
than one area of the address map. Table 12-5 shows memory bank sizes from 256 Kbytes up to 16 Gbytes.

Table 12-5. 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

12.3.1.2.2 Option Registers (ORx)—GPCM Mode

Table 12-6 shows the bit fields for OR*x* when the corresponding BR*x*[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 12-3. Option Registers in GPCM Mode

Table 12-6 describes OR*x* fields for GPCM mode.

Table 12-6. ORx—GPCM Field Descriptions

Bits	Name	Description
0–16	AM	GPCM address mask. Masks corresponding BR*x* 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 BR*x* 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 $\overline{\text{LCS}}$x and $\overline{\text{LWE}}$ 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 $\overline{\text{LCS}}$x and $\overline{\text{LWE}}$ are negated normally. 1 $\overline{\text{LCS}}$x and $\overline{\text{LWE}}$ are negated earlier depending on the value of LCRR[CLKDIV].

LCRR[CLKDIV]	CSNT	Meaning
x	0	$\overline{\text{LCS}}$x and $\overline{\text{LWE}}$ are negated normally.
2	1	$\overline{\text{LCS}}$x and $\overline{\text{LWE}}$ are negated normally.
4 or 8	1	$\overline{\text{LCS}}$x and $\overline{\text{LWE}}$ are negated quarter of a bus clock cycle earlier.

Table 12-6. ORx—GPCM Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>21–22</td><td>ACS</td><td>Address to chip-select setup. Determines the delay of the LCSx assertion relative to the address change when the external memory access is handled by the GPCM. At system reset, OR0[ACS] = 11
<table>
<tr><th>LCRR[CLKDIV]</th><th>Value</th><th>Meaning</th></tr>
<tr><td>x</td><td>00</td><td>LCSx is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT=0.</td></tr>
<tr><td>x</td><td>01</td><td>Reserved.</td></tr>
<tr><td>2</td><td>10</td><td>LCSx is output a half bus clock cycle after the address lines.</td></tr>
<tr><td>2</td><td>11</td><td>LCSx is output a half bus clock cycle after the address lines.</td></tr>
<tr><td>4 or 8</td><td>10</td><td>LCSx is output a quarter bus clock cycle after the address lines.</td></tr>
<tr><td>4 or 8</td><td>11</td><td>LCSx is output a half bus clock cycle after the address lines.</td></tr>
</table></td></tr>
<tr><td>23</td><td>XACS</td><td>Extra address to chip-select setup. Setting this bit increases the delay of the LCSx assertion relative to the address change when the external memory access is handled by the GPCM. After a system reset, OR0[XACS] = 0.
0 Address to chip-select setup is determined by ORx[ACS] and LCRR[CLKDIV].
1 Address to chip-select setup is extended (see Table 12-24 and Table 12-24 for LCRR[CLKDIV]=4 or 8, Table 12-25 and Table 12-26 for LCRR[CLKDIV]=2).</td></tr>
<tr><td>24–27</td><td>SCY</td><td>Cycle length in bus clocks. Determines the number of wait states inserted in the bus cycle, when the GPCM handles the external memory access. Thus it is the main parameter for determining cycle length. The total cycle length depends on other timing attribute settings. After a system reset, OR0[SCY] = 1111.
0000 No wait states
0001 1 bus clock cycle wait state
...
1111 15 bus clock cycle wait states</td></tr>
<tr><td>28</td><td>SETA</td><td>External address termination.
0 Access is terminated internally by the memory controller unless the external device asserts LGTA earlier to terminate the access.
1 Access is terminated externally by asserting the LGTA external pin. (Only LGTA can terminate the access).</td></tr>
<tr><td>29</td><td>TRLX</td><td>Timing relaxed. Modifies the settings of timing parameters for slow memories or peripherals.
0 Normal timing is generated by the GPCM.
1 Relaxed timing on the following parameters:
• Adds an additional cycle between the address and control signals (only if ACS is not equal to 00).
• Doubles the number of wait states specified by SCY, providing up to 30 wait states.
• Works in conjunction with EHTR to extend hold time on read accesses.
• LCSx (only if ACS is not equal to 00) and LWE signals are negated one cycle earlier during writes.</td></tr>
</table>

Table 12-6. ORx—GPCM Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>30</td><td>EHTR</td><td>Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.
<table>
<tr><th>TRLX</th><th>EHTR</th><th>Meaning</th></tr>
<tr><td>0</td><td>0</td><td>The memory controller generates normal timing. No additional cycles are inserted.</td></tr>
<tr><td>0</td><td>1</td><td>1 idle clock cycle is inserted.</td></tr>
<tr><td>1</td><td>0</td><td>4 idle clock cycles are inserted.</td></tr>
<tr><td>1</td><td>1</td><td>8 idle clock cycles are inserted.</td></tr>
</table></td></tr>
<tr><td>31</td><td>EAD</td><td>External address latch delay. Allow extra bus clock cycles when using external address latch (LALE).
0 No additional bus clock cycles (LALE asserted for one bus clock cycle only)
1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).</td></tr>
</table>

12.3.1.2.3 Option Registers (ORx)—UPM Mode

Table 12-7 shows the bit fields for OR*x* when the corresponding BR*x*[MSEL] selects a UPM machine.

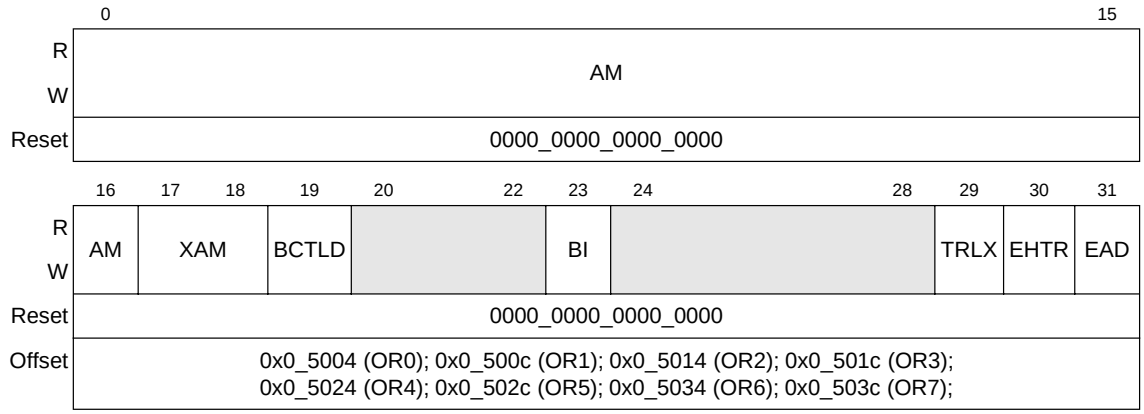


Figure 12-4. Option Registers in UPM Mode

Table 12-7 describes BR*x* fields for UPM mode.

Table 12-7. OR*x*—UPM Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0–16</td><td>AM</td><td>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.</td></tr>
<tr><td>17–18</td><td>XAM</td><td>Extended address mask. Masks the corresponding XBA bits in the BRx register, effectively extending the address mask (AM) by 2 bits.</td></tr>
<tr><td>19</td><td>BCTLD</td><td>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.</td></tr>
<tr><td>20–22</td><td>—</td><td>Reserved</td></tr>
<tr><td>23</td><td>BI</td><td>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.</td></tr>
<tr><td>24–28</td><td>—</td><td>Reserved</td></tr>
<tr><td>29</td><td>TRLX</td><td>Timing relaxed. Works in conjunction with EHTR to extend hold time on read accesses.</td></tr>
<tr><td>30</td><td>EHTR</td><td>Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.
<table>
<tr><th>TRLX</th><th>EHTR</th><th>Meaning</th></tr>
<tr><td>0</td><td>0</td><td>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>
</table>
</td></tr>
<tr><td>31</td><td>EAD</td><td>External address latch delay. Allow extra bus clock cycles when using external address latch (LALE).
0 No additional bus clock cycles (LALE asserted for one bus clock cycle only)
1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).</td></tr>
</table>

12.3.1.2.4 Option Registers (ORx)—SDRAM Mode

Figure 12-5 shows the bit fields for OR*x* when the corresponding BR*x*[MSEL] selects the SDRAM machine.

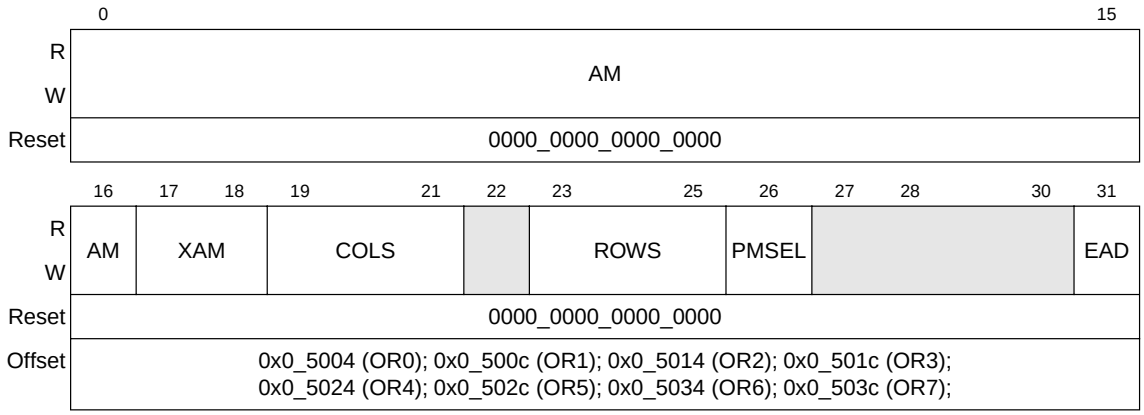


Figure 12-5. Option Registers in SDRAM Mode

Table 12-8 describes BR*x* fields for SDRAM mode.

Table 12-8. 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 12-8. 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]).

12.3.1.3 UPM Memory Address Register (MAR)

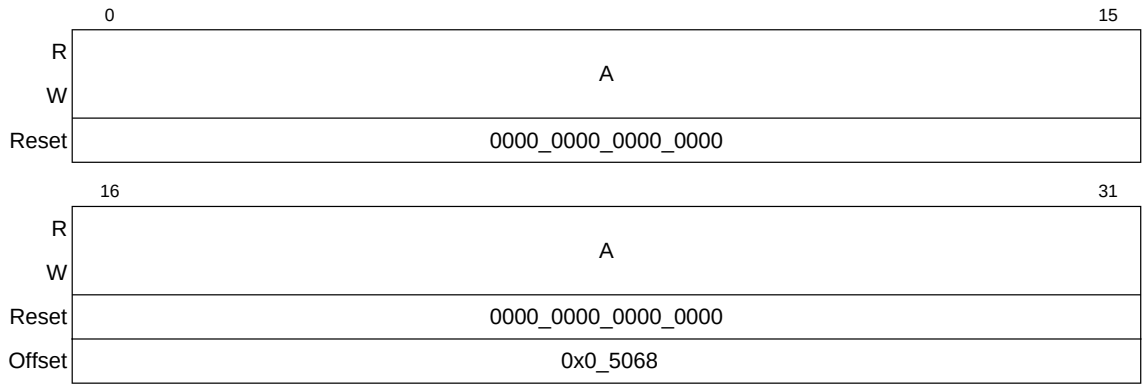


Figure 12-6. UPM Memory Address Register

Table 12-9 describes MAR fields.

Table 12-9. 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.

12.3.1.4 UPM Mode Registers (M*x*MR)

The UPM machine mode registers (MAMR, MBMR and MCMR) contain the configuration for the three UPMs.

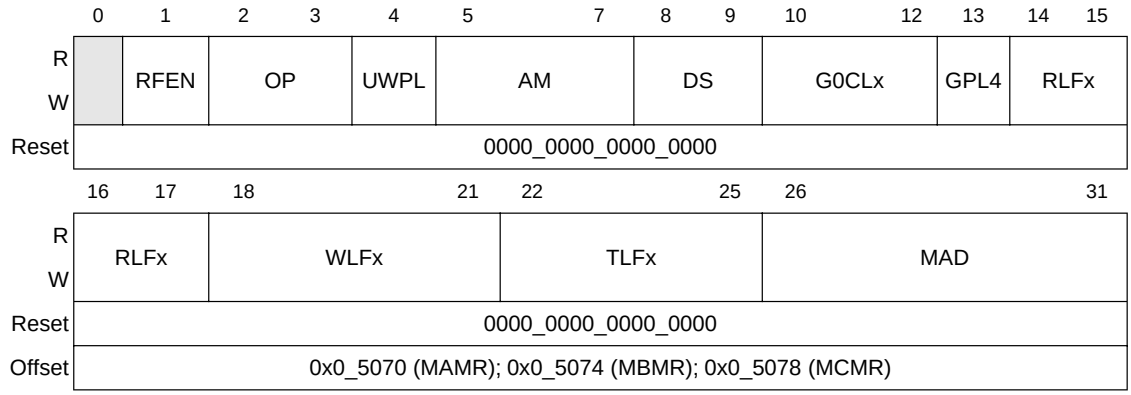


Figure 12-7. UPM Mode Registers

Table 12-10 describes UPM mode fields.

Table 12-10. M*x*MR Field Descriptions

Bits	Name	Description
0	—	Reserved
1	RFEN	Refresh enable. Indicates that the UPM needs refresh services. This bit must be set for UPMA (refresh executor) if refresh services are required on any UPM assigned chip selects. If MAMR[RFEN]=0, no refresh services 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 12-10. M*x*MR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>5–7</td><td>AM</td><td>Address multiplex size. Determines how the address of the current memory cycle can be output on the address pins. This field is needed when interfacing with devices requiring row and column addresses multiplexed on the same pins.
<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="8">Reserved</td></tr>
<tr><td>111</td><td colspan="8">Reserved</td></tr>
</table></td></tr>
<tr><td>8–9</td><td>DS</td><td>Disable timer period. Guarantees a minimum time between accesses to the same memory bank controlled by 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</td></tr>
<tr><td>10–12</td><td>G0CLx</td><td>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</td></tr>
<tr><td>13</td><td>GPL4</td><td>LGPL4 output line disable. Determines how the LGPL4/UPWAIT pin is controlled by the corresponding bits in the UPMx array. See Table 12-30 on page 12-73.
<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></td></tr>
</table>

Table 12-10. M*x*MR Field Descriptions (continued)

Bits	Name	Description
14–17	RLFx	Read loop field. Determines the number of times a loop defined in the UPM*x* 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.

12.3.1.5 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.

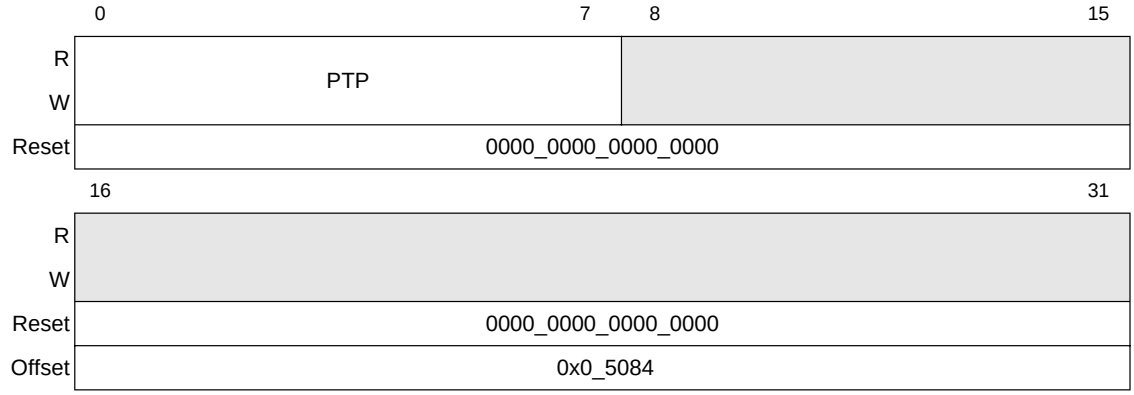


Figure 12-8. Memory Refresh Timer Prescaler Register

Table 12-11 describes MRTPR fields.

Table 12-11. 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

12.3.1.6 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.

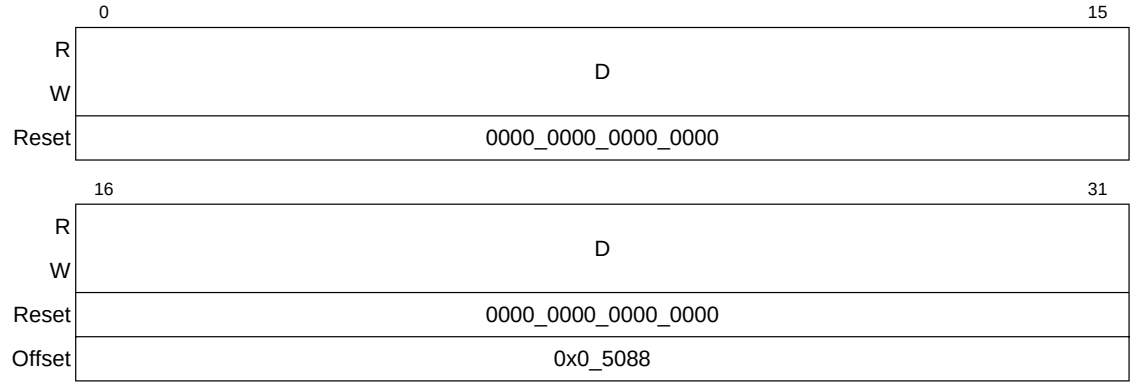


Figure 12-9. UPM Data Register

Table 12-12 describes MDR fields.

Table 12-12. 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).

12.3.1.7 SDRAM Machine Mode Register (LSDMR)

The local bus SDRAM mode register (LSDMR) is used to configure operations pertaining to SDRAM.

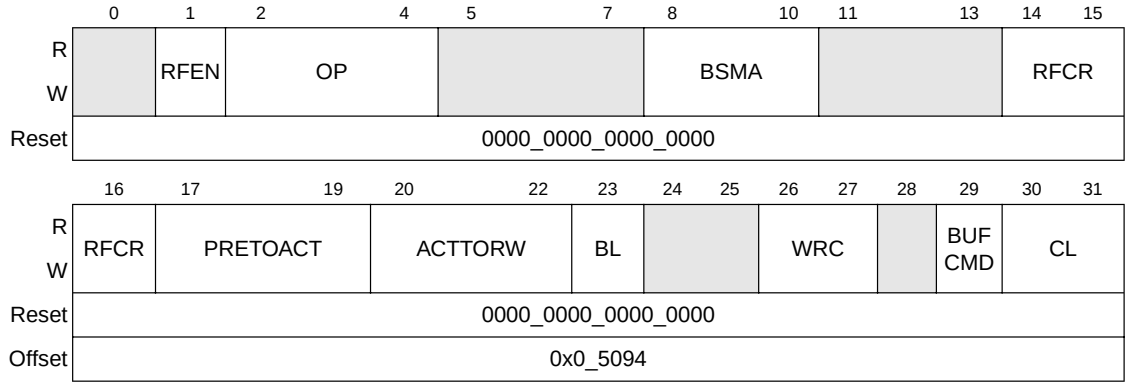


Figure 12-10. SDRAM Machine Mode Register (LSDMR)

Table 12-12 describes LSDMR fields.

Table 12-13. 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 12-13. LSDMR Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>2–4</td><td>OP</td><td>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></td></tr>
<tr><td>5–7</td><td>—</td><td>Reserved</td></tr>
<tr><td>8–10</td><td>BSMA</td><td>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</td></tr>
<tr><td>11–13</td><td>—</td><td>Reserved</td></tr>
<tr><td>14–16</td><td>RFCR</td><td>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</td></tr>
<tr><td>17–19</td><td>PRETOACT</td><td>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</td></tr>
</table>

Table 12-13. 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 is16. 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 $\overline{\text{LCS}}$x, LCKE, LALE and $\overline{\text{LSDDQM}[0:3]}$ to be asserted for LCRR[BUFCMDC] cycles, instead of one. 0 Normal timing for the control lines 1 All control lines except $\overline{\text{LCS}}$x are asserted for the number of cycles specified by LCRR[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 LCRR[ECL]. See Table 12-22 on page 12-36. 01 1 10 2 11 3

12.3.1.8 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 (M*x*MR[RFEN] =1). Each time the timer expires, a qualified bank generates a refresh request using the selected UPM. The qualified banks rotate their requests.

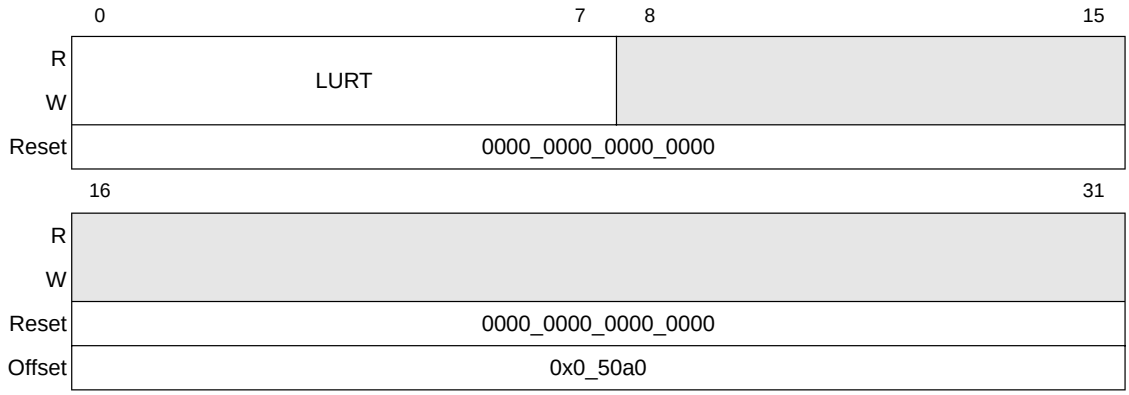


Figure 12-11. UPM Refresh Timer (LURT)

Table 12-14 describes LURT fields.

Table 12-14. 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.

12.3.1.9 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.

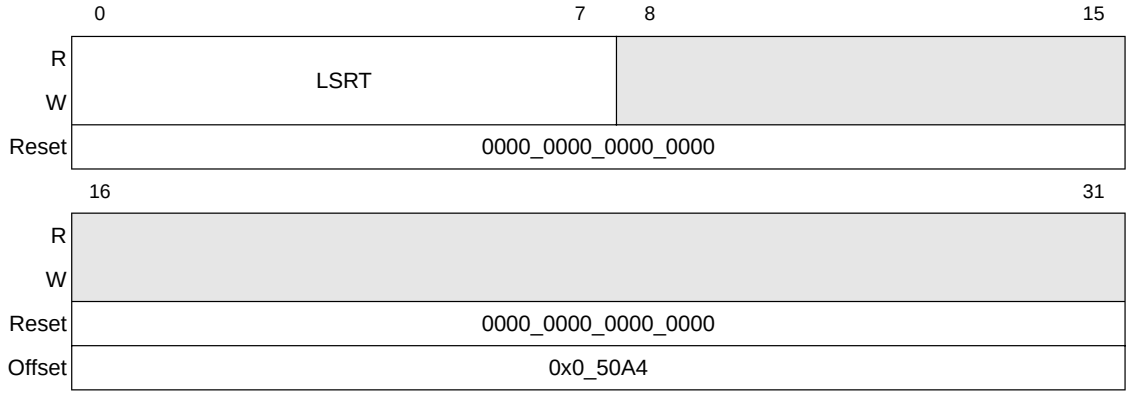


Figure 12-12. LSRT SDRAM Refresh Timer

Table 12-15 describes LSRT fields.

Table 12-15. 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: $\text{TimerPeriod} = \frac{\text{LSRT}}{\left(\frac{\text{Fsystemclock}}{\text{MRTPR[PTP]}}\right)}$ 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

12.3.1.10 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.

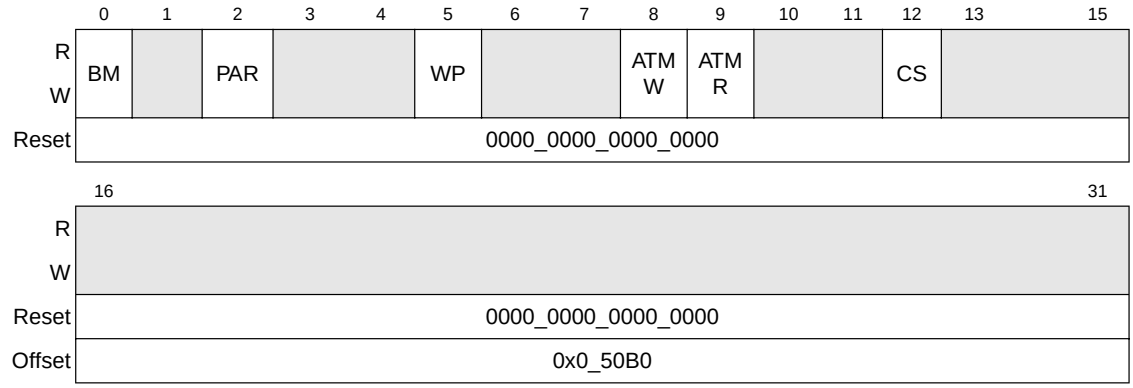


Figure 12-13. Transfer Error Status Register

Table 12-16 describes LTESR fields.

Table 12-16. LTESR Field Descriptions

Bits	Name	Description
0	BM	Bus monitor time-out. 0 No local bus monitor time-out occurred 1 Local bus monitor time-out occurred. No data beat was acknowledged on the bus within LBCR[BMT] x 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[BNK] indicates which memory controller bank was accessed.
3–4	—	Reserved
5	WP	Write protect error 0 No write protect error occurred 1 A write was attempted to a local bus memory region that was defined as read-only in the memory controller. Usually, in this case, a bus monitor time-out will occur.
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

12.3.1.11 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.

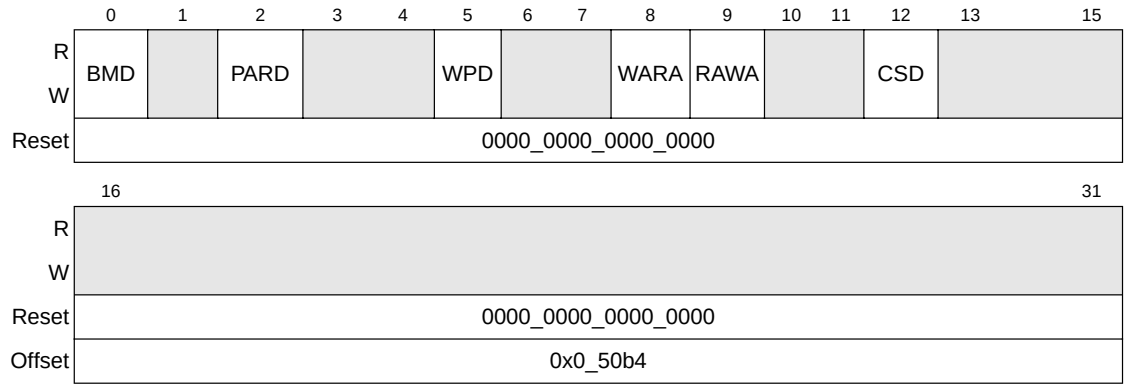


Figure 12-14. Transfer Error Check Disable Register

Table 12-17 describes LTEDR fields.

Table 12-17. 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

12.3.1.12 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.

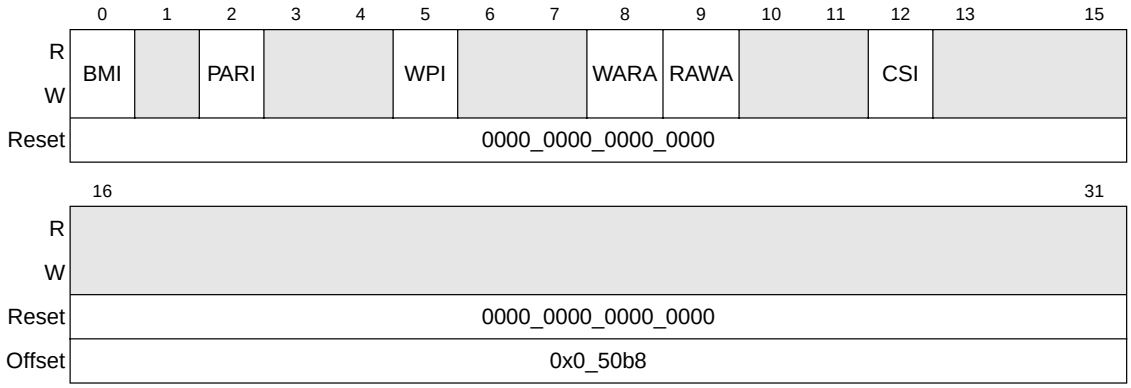


Figure 12-15. Transfer Error Interrupt Enable Register

Table 12-18 describes LTEIR fields.

Table 12-18. 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 12-18. 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

12.3.1.13 Transfer Error Attributes Register (LTEATR)

Figure 12-16 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.

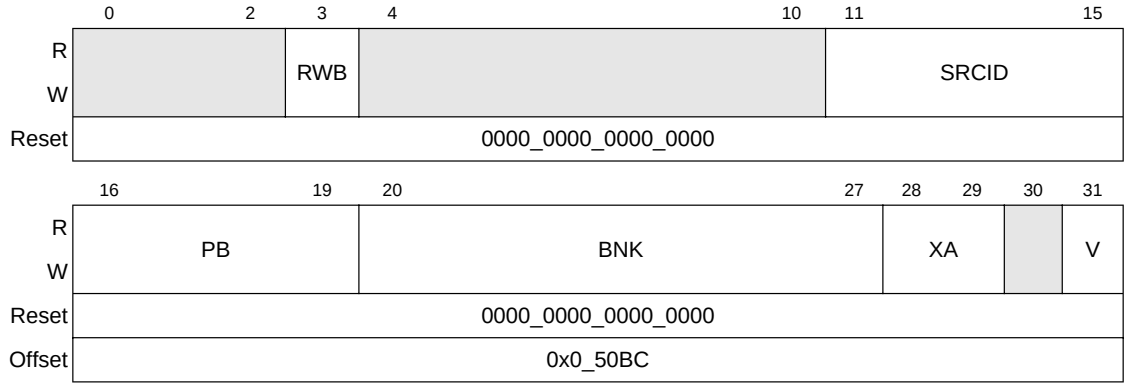


Figure 12-16. Transfer Error Attributes Register

Table 12-19 describes LTEATR fields.

Table 12-19. 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 12-19. 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.

12.3.1.14 Transfer Error Address Register (LTEAR)

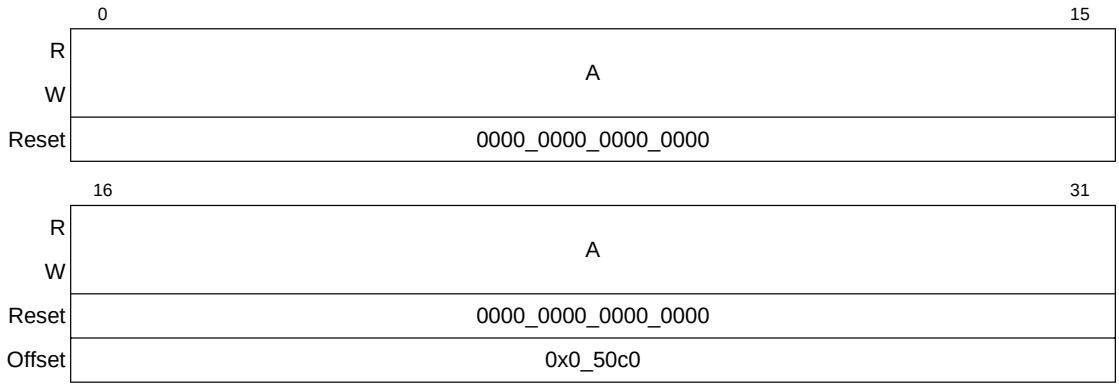


Figure 12-17. Transfer Error Address Register

Table 12-20 describes LTEAR fields.

Table 12-20. 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.

12.3.1.15 Local Bus Configuration Register (LBCR)

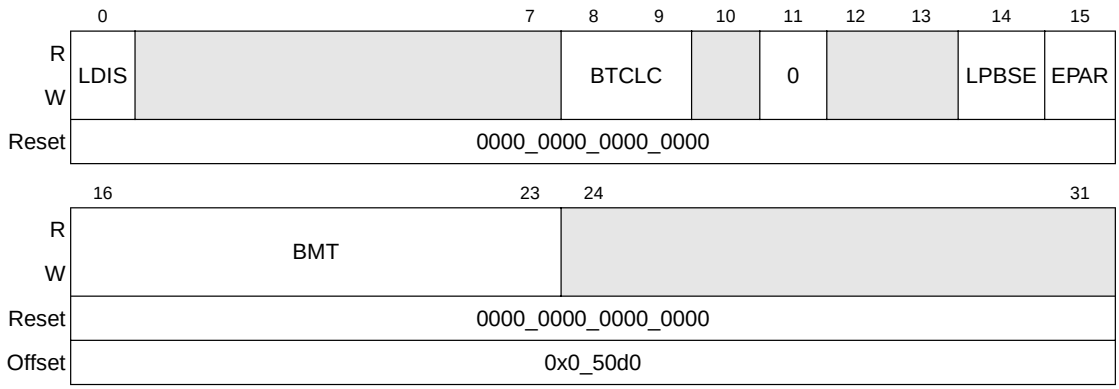


Figure 12-18. Local Bus Configuration Register

Table 12-21 describes LBCR fields.

Table 12-21. LBCR Field Descriptions

Bits	Name	Description
0	LDIS	Local bus disable. 0 Local bus is enabled. 1 Local bus is disabled. No internal transactions will be acknowledged.
1–7	—	Reserved
8–9	BCTLC	Defines the use of LBCTL. 00 LBCTL is used as W/$\overline{R}$ control for GPCM or UPM accesses (buffer control). 01 LBCTL is used as $\overline{LOE}$ for GPCM accesses only. 10 LBCTL is used as $\overline{LWE}$ for GPCM accesses only. 11 Reserved.
10–11	—	Reserved
12–13	—	Reserved
14	LPBSE	Enables parity byte select on $\overline{LGTA}$/LGPL4/UPWAIT/LPBSE pin. 0 Parity byte select is disabled. $\overline{LGTA}$/LGPL4/UPWAIT/LPBSE pin is available for memory control as LGPL4 (output) or $\overline{LGTA}$/UPWAIT (input). 1 Parity byte select is enabled. $\overline{LGTA}$/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

12.3.1.16 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.

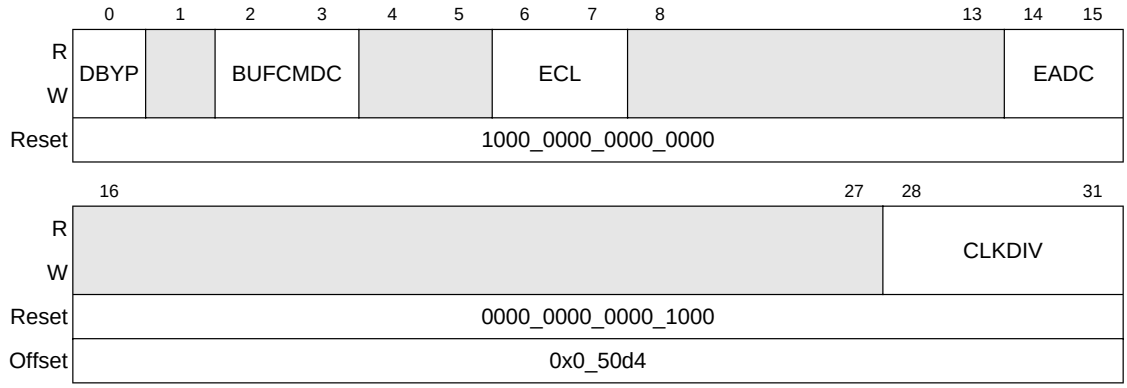


Figure 12-19. Clock Ratio Register (LCRR)

Table 12-22 describes LCRR fields.

Table 12-22. 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 12-22. 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

12.4 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.

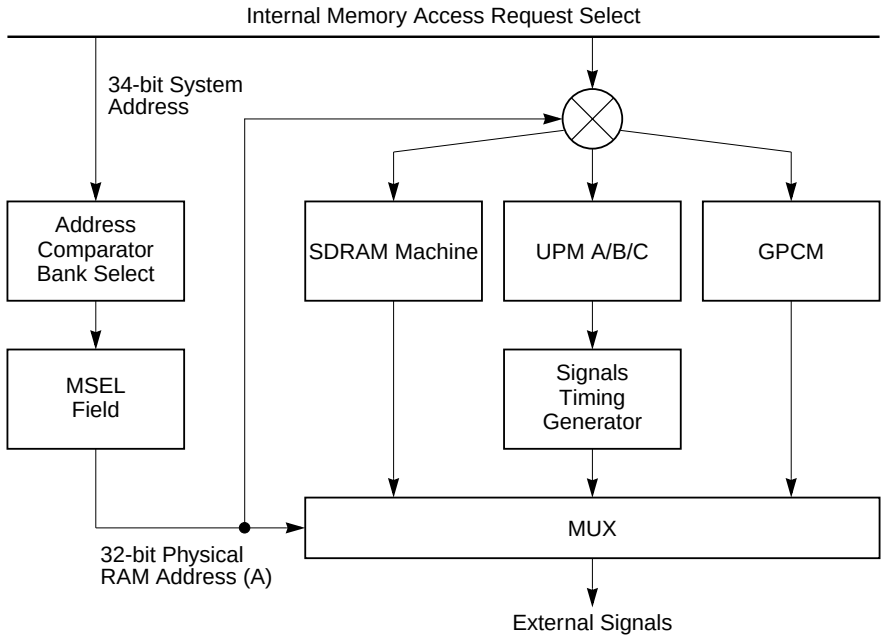


Figure 12-20. Basic Operation of Memory Controllers in LBC

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 12-20. 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.

12.4.1 Basic Architecture

12.4.1.1 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).

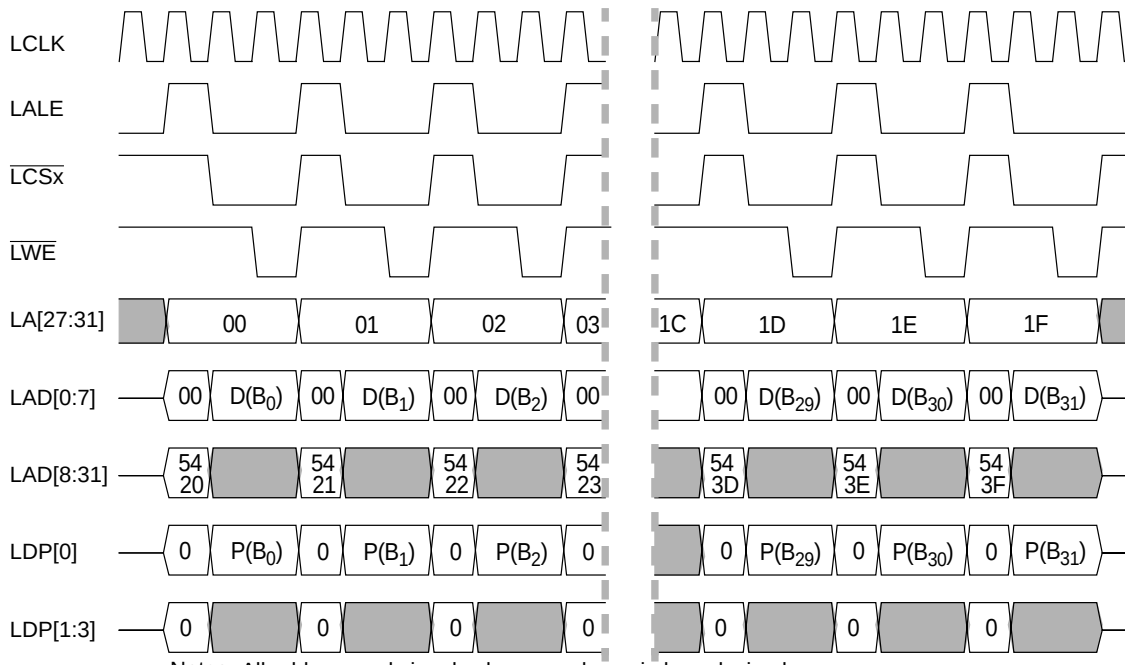
12.4.1.2 External Address Latch Enable Signal (LALE)

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 12-21 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.



Notes: All address and signal values are shown in hexadecimal.
D(Bk) = kth of 32 data bytes, P(Bk) = parity bit of kth data byte.

Figure 12-21. Example of 8-bit GPCM writing 32 bytes to address 0x5420.

12.4.1.3 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 12-22. 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.

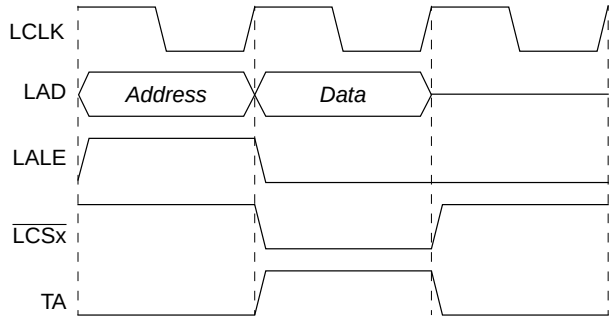


Figure 12-22. Basic LBC Bus Cycle with LALE, TA, and $\overline{\text{LCSx}}$

12.4.1.4 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 ORx[BCTLD]. Access to an SDRAM machine controlled bank does not activate the LBCTL control. The LBCTL control can be further configured by LBCR[BCTLC] to act as an extra $\overline{\text{LWE}}$ or an extra $\overline{\text{LOE}}$ signal when in GPCM mode.

If LBCTL is configured as a data buffer control (LBCR[BCTLC] = 00), the signal is asserted (high) on the rising edge of the bus clock on the first cycle of the memory controller operation, coincident with LALE. If the access is a write, LBCTL remains high for the whole duration. However, if the access is a read, LBCTL is negated (low) with the negation of LALE so that the memory device is able to drive the bus. If back-to-back read accesses are pending, LBCTL is asserted (high) one bus clock cycle before the next transaction starts (that is, one bus clock cycle before LALE) to allow a whole bus cycle for the bus to turn around before the next address is driven.

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.

12.4.1.5 Atomic Operation

The LBC supports the following kinds of atomic bus operations (set by BR*x*[ATOM]):

- Read-after-write (RAWA). When a write access hits a memory bank in which 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).

12.4.1.6 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).

12.4.1.7 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.

12.4.2 General-Purpose Chip-Select Machine (GPCM)

The GPCM allows a minimal glue logic and flexible interface to SRAM, EPROM, FEPROM, ROM devices, and external peripherals. The GPCM contains two basic configuration register groups—BR*x* and OR*x*.

Figure 12-23 shows a simple connection between an 8-bit port size SRAM device and the local bus controller in GPCM mode. Byte-write enable signals ($\overline{\text{LWE}}$) are available for each byte written to memory. Also, the output enable signal ($\overline{\text{LOE}}$) is provided to minimize

external glue logic. On system reset, a global (boot) chip-select is available that provides a boot ROM chip-select ($\overline{\text{LCS0}}$) prior to the system being fully configured.

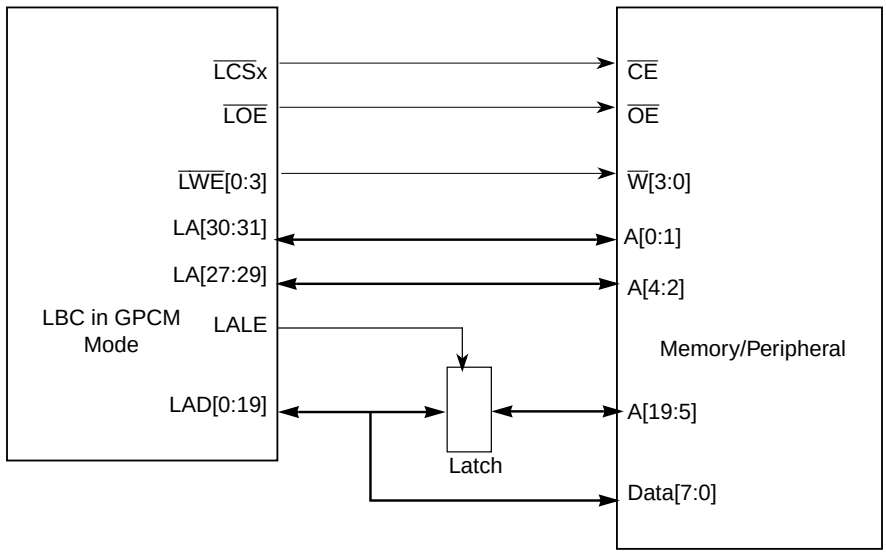


Figure 12-23. Local Bus to GPCM Device Interface

Figure 12-24 shows $\overline{\text{LCS}}$ as defined by the setup time required between the address lines and $\overline{\text{CE}}$. The user can configure ORx[ACS] to specify $\overline{\text{LCS}}$ to meet this requirement.

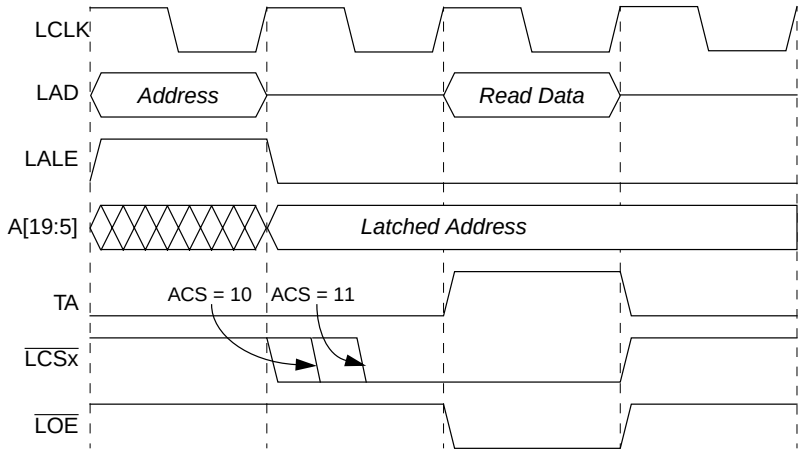


Figure 12-24. GPCM Basic Read Timing
(XACS = 0, ACS = 1x, TRLX = 0, CLKDIV = 4,8)

12.4.2.1 Timing Configuration

If BRx[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

LCRR[CLKDIV] = 4 or LCRR[CLKDIV] = 8. Table 12-24 shows the signal behavior and system response for a read access with LCRR[CLKDIV] = 4 or LCRR[CLKDIV] = 8. Table 12-25 and Table 12-26 show the write and read signal behavior, respectively, when LCRR[CLKDIV] = 2.

Table 12-23. 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 12-24. 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 12-25. 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 12-26. 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.

12.4.2.2 Chip-Select Assertion Timing

The banks selected to work with the GPCM, support an option to drive the $\overline{\text{LCS}x}$ signal with different timings (with respect to the external address/data bus). $\overline{\text{LCS}x}$ can be driven in any of the following ways:

- 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 12-24 shows two chip-select assertion timings for the case LCRR[CLKDIV] = 4 or 8. If LCRR[CLKDIV] = 2, $\overline{\text{LCS}x}$ asserts identically for ORx[ACS] = 10 or 11.

12.4.2.2.1 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 OR*x*[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 $\overline{\text{LWEx}}$ 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.

12.4.2.2.2 Chip-Select and Write Enable Negation Timing

Figure 12-23 shows a basic connection between the local bus and a static memory device. In this case, $\overline{\text{LCS}x}$ 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.

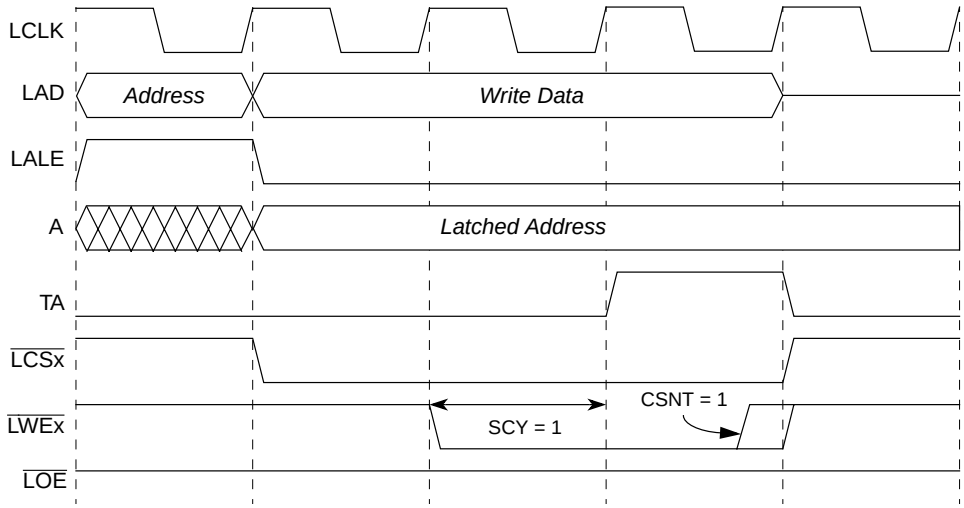


Figure 12-25. GPCM Basic Write Timing
(XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0, CLKDIV = 4, 8)

As Figure 12-25 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). ORx[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 LCRR[CLDIV] = 4 or 8. For example, when ACS = 00 and CSNT = 1, $\overline{\text{LWEx}}$ is negated one quarter of a clock earlier, as shown in Figure 12-25. If LCRR[CLDIV] = 2, $\overline{\text{LWEx}}$ is negated either coincident with $\overline{\text{LCSx}}$ or one cycle earlier.

12.4.2.2.3 Relaxed Timing

OR*x*[TRLX] is provided for memory systems that require more relaxed timing between signals. Setting TRLX = 1 has the following effect on timing:

- An additional bus cycle is added between the address and control signals (but only if ACS is not equal to 00).
- The number of wait states specified by SCY is doubled, providing up to 30 wait states.
- The extended hold time on read accesses (EHTR) is extended further.
- $\overline{\text{LCSx}}$ 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.

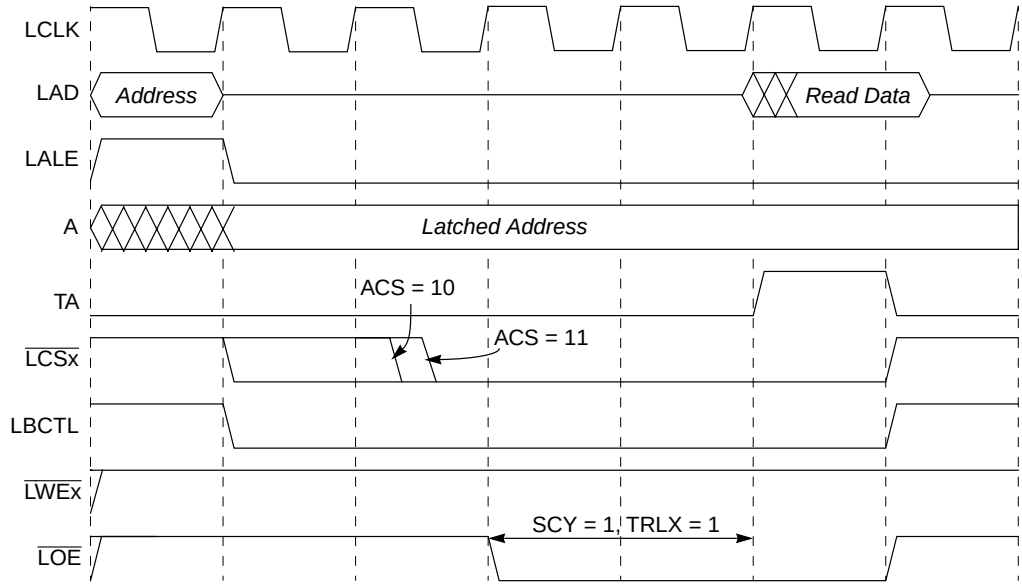


Figure 12-26. GPCM Relaxed Timing Read
(XACS = 0, ACS = 1x, SCY = 1, CSNT = 0, TRLX = 1, CLKDIV = 4, 8)

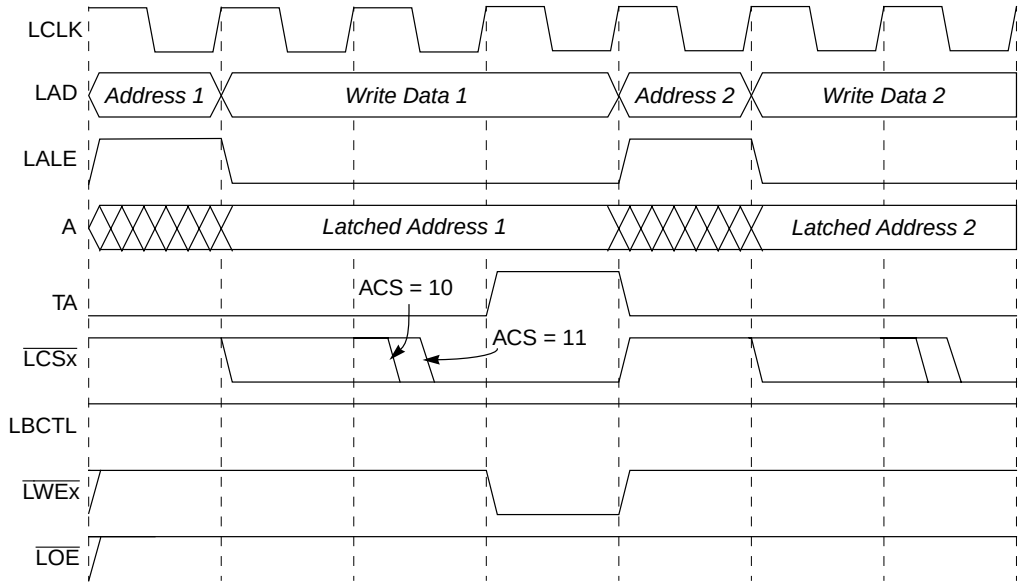


Figure 12-27. GPCM Relaxed Timing Back-to-Back Writes
(XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1, CLKDIV = 4, 8)

Relaxed timing read and write transactions are illustrated in Figure 12-26 and Figure 12-27. 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 12-27 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 ACS ≠ 00, $\overline{\text{LCSx}}$ is also negated one clock earlier.

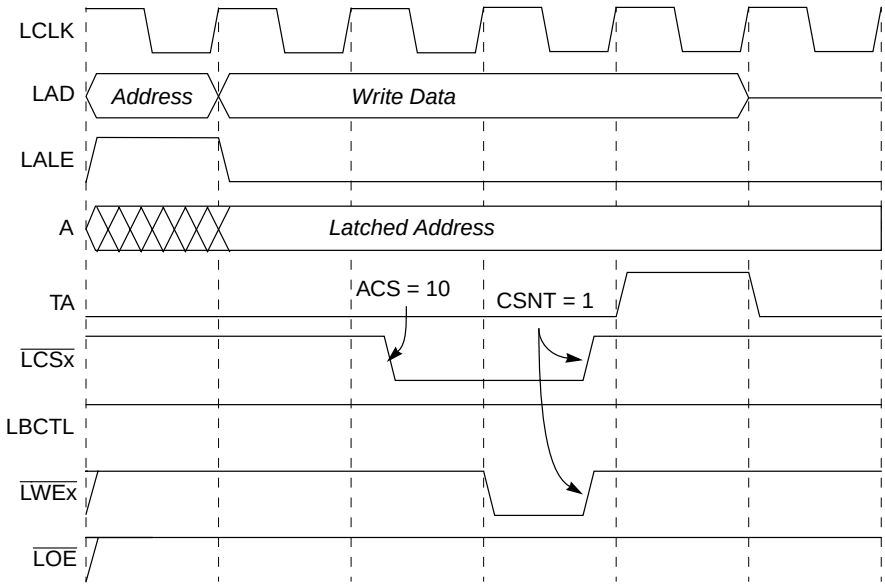


Figure 12-28. GPCM Relaxed Timing Write
(XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1, CLKDIV = 4, 8)

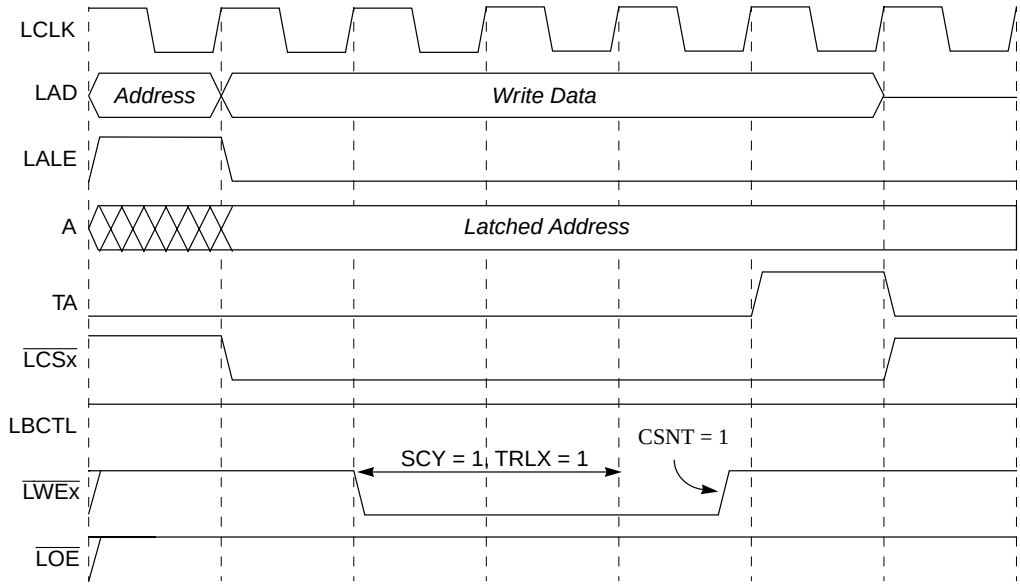


Figure 12-29. GPCM Relaxed Timing Write
(XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1, CLKDIV = 4, 8)

12.4.2.2.4 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 $\overline{\text{LCS}x}$ is asserted or coinciding with $\overline{\text{LCS}x}$ (if XACS = 1 and ACS = 10 or ACS = 11). Accordingly, assertion of $\overline{\text{LOE}}$ can be delayed (along with the assertion of $\overline{\text{LCS}x}$) by programming TRLX = 1. $\overline{\text{LOE}}$ negates on the rising clock edge coinciding with $\overline{\text{LCS}x}$ negation

12.4.2.2.5 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 OR*x*[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 12-6 in addition to any existing bus turnaround cycle.

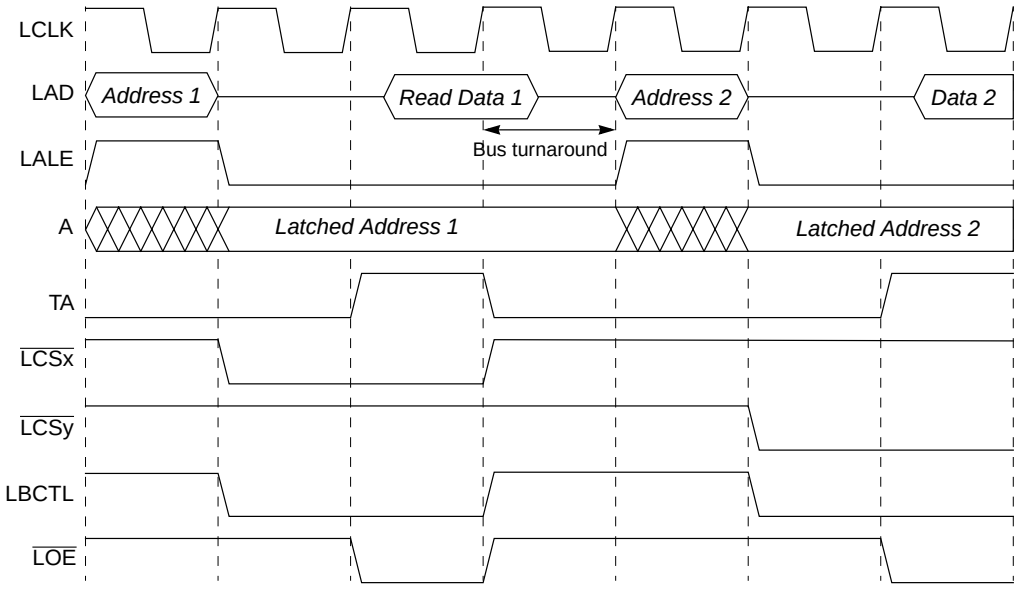


Figure 12-30. GPCM Read Followed by Read (TRLX = 0, EHTR = 0, Fastest Timing)

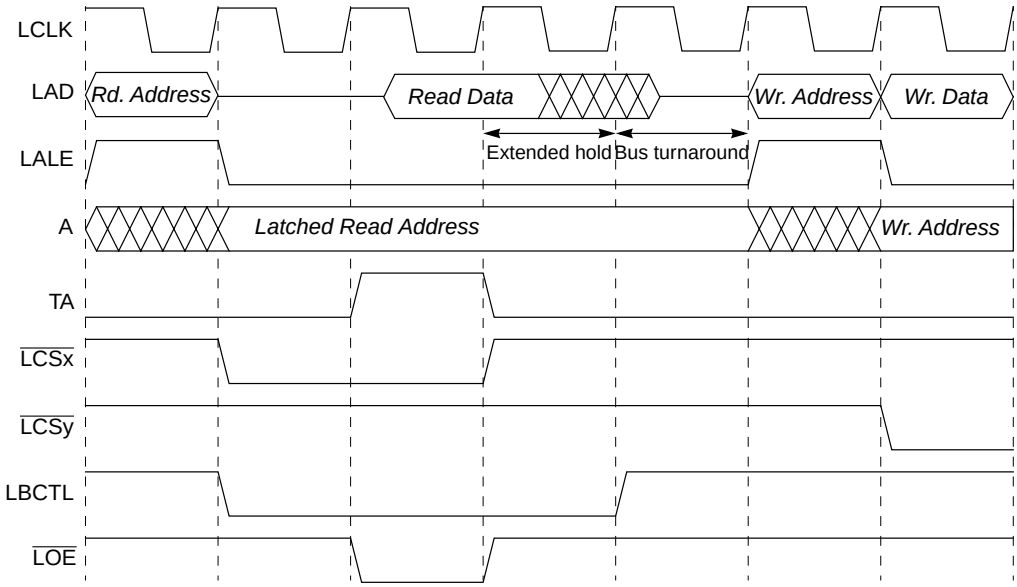


Figure 12-31. GPCM Read Followed by Write (TRLX = 0, EHTR = 1, 1-cycle Extended Hold Time on Reads)

12.4.2.3 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 ORx[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 occusr 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 ORx[SETA]. Asserting $\overline{\text{LGTA}}$ always terminates an access, even if OR*x*[SETA] = 0 (internal transfer acknowledge generation), but it is the only means by which an access can be terminated if ORx[SETA] = 1. The timing of $\overline{\text{LGTA}}$ is illustrated by the example in Figure 12-32.

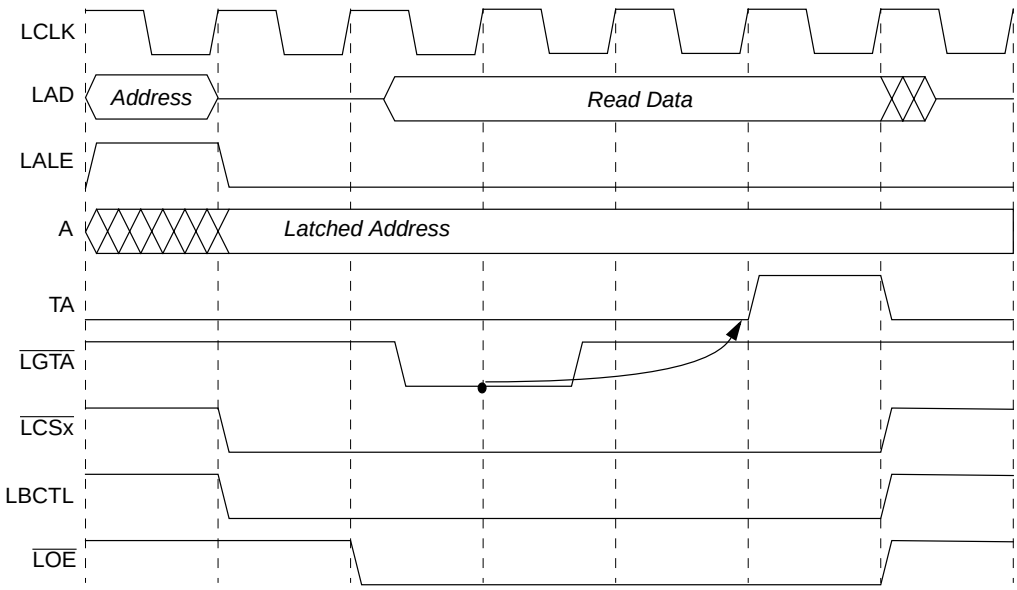


Figure 12-32. External Termination of GPCM Access

12.4.2.4 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 12-27 describes the initial values of the boot bank in the memory controller.

Table 12-27. 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

12.4.3 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.

12.4.3.1 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 12-33 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 signa, while the remaining address bits (except A10) are latched from LAD[20:26].

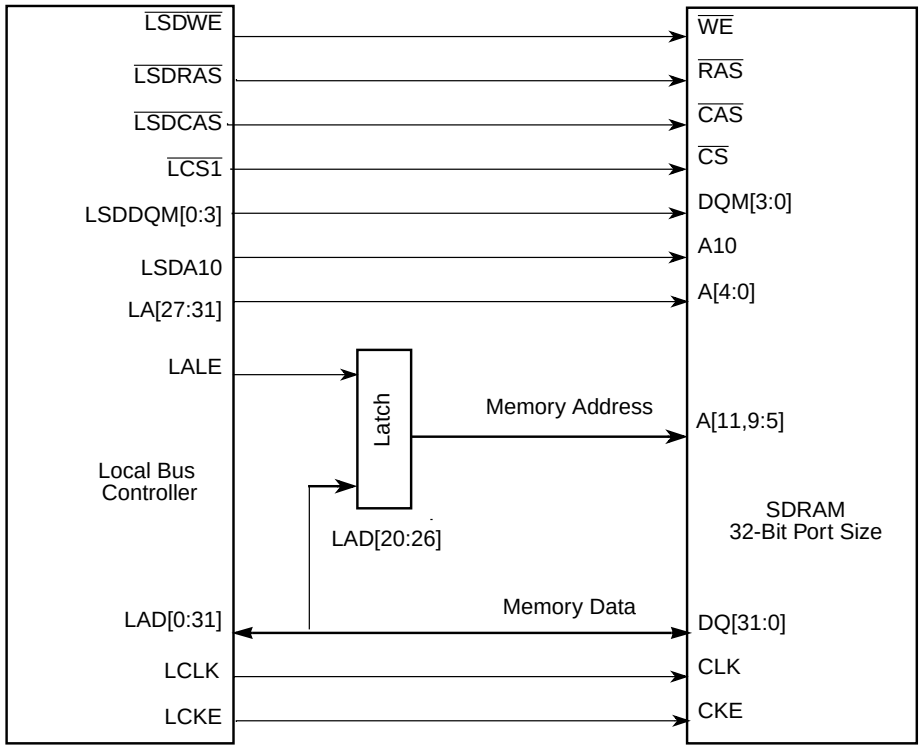


Figure 12-33. Connection to a 32-Bit SDRAM with 12 Address Lines

12.4.3.2 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.

12.4.3.3 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 12-28. 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.

12.4.3.4 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.

12.4.3.5 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.

12.4.3.6 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 12-34 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 (Ax in Figure 12-34) 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).

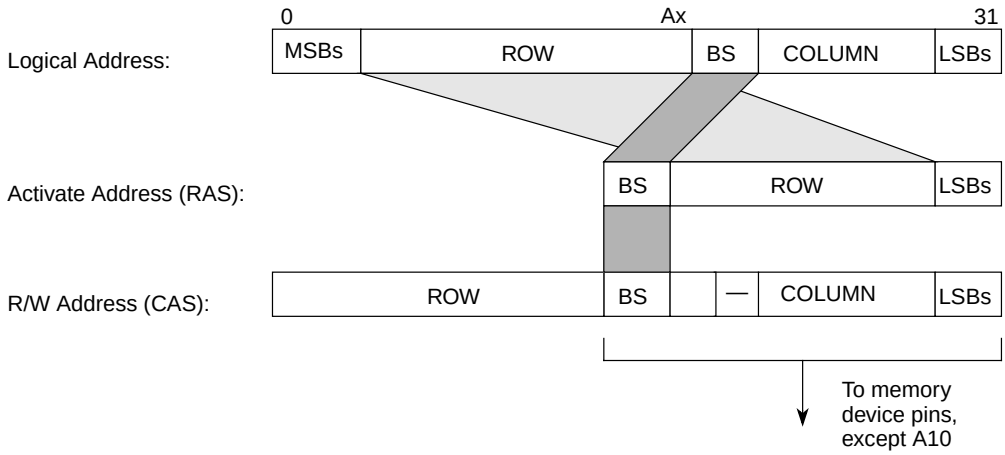


Figure 12-34. SDRAM Address Multiplexing

Note that during normal operation (read/write), a full 32-bit address which incluades 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.

12.4.3.7 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 OR*x* 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.

12.4.3.7.1 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.

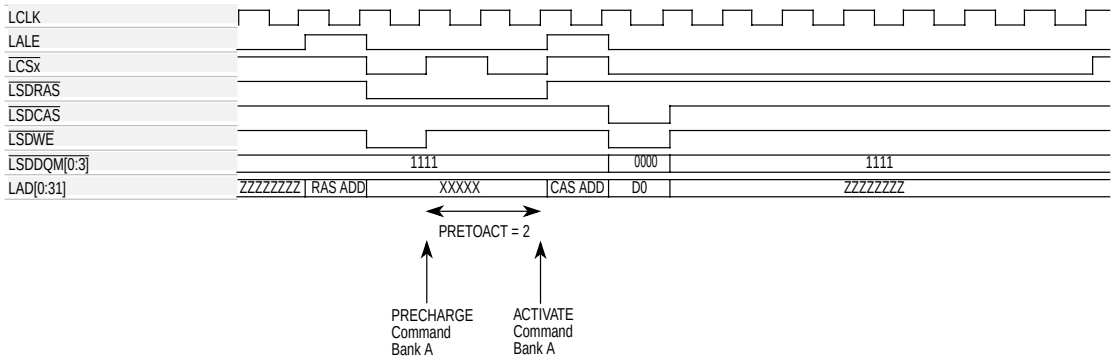


Figure 12-35. PRETOACT = 2 (2 Clock Cycles)

12.4.3.7.2 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.

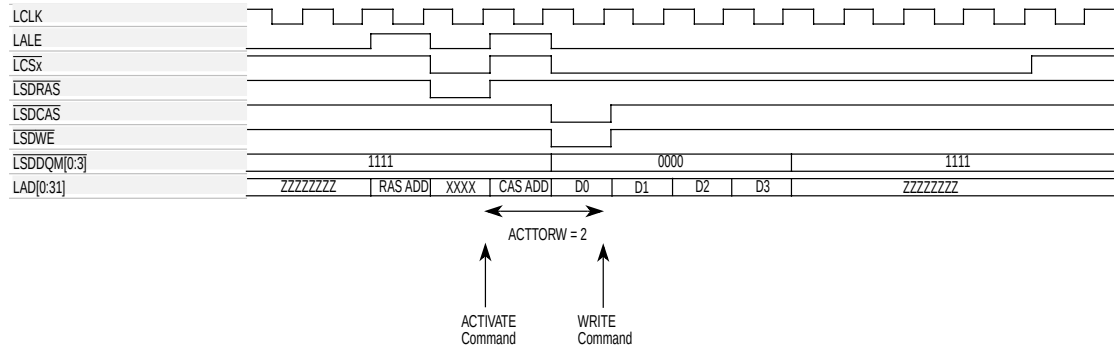


Figure 12-36. ACTTORW = 2 (2 Clock Cycles)

12.4.3.7.3 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.

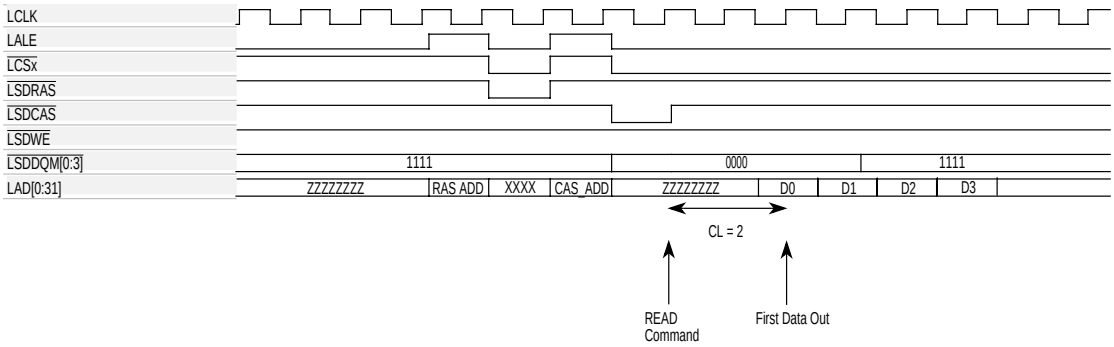


Figure 12-37. CL = 2 (2 Clock Cycles)

12.4.3.7.4 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.

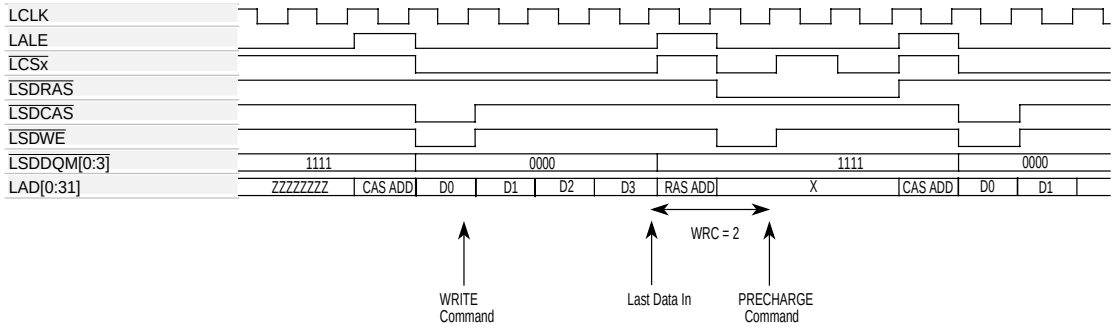


Figure 12-38. WRC = 2 (2 Clock Cycles)

12.4.3.7.5 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.

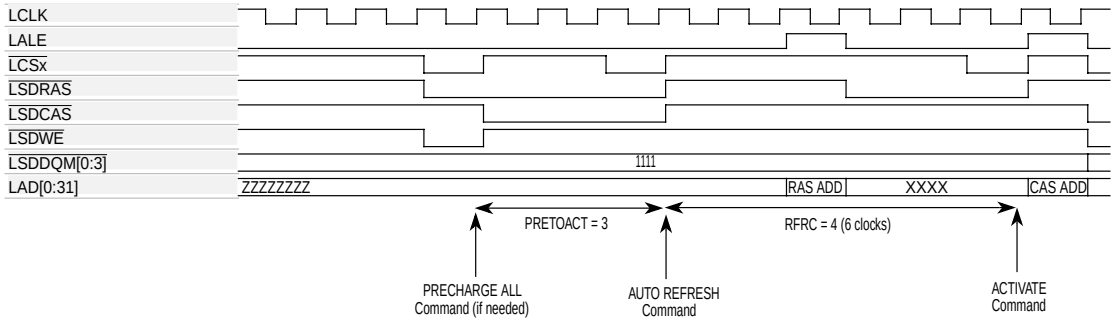


Figure 12-39. RFRC = 4 (6 Clock Cycles)

12.4.3.7.6 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, LSDMR[BUFCMD] should be set. Setting this bit causes the memory controller to add LCRR[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.

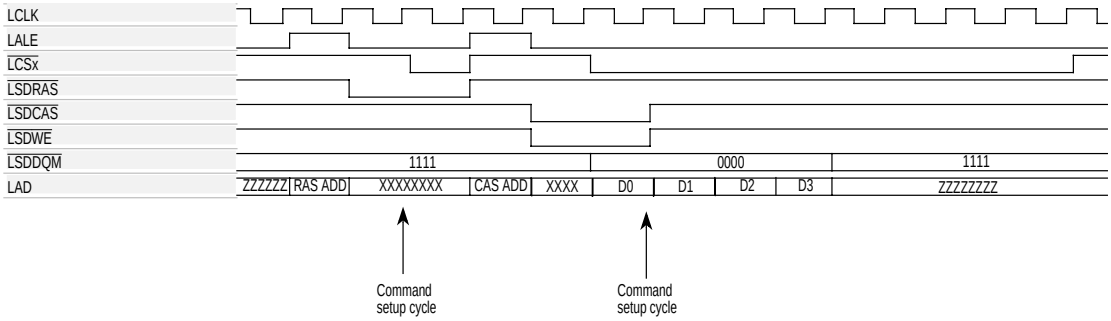


Figure 12-40. BUFCMD = 1, LCRR[BUFCMDC] = 2

12.4.3.8 SDRAM Interface Timing

The following figures show SDRAM timing for various types of accesses.

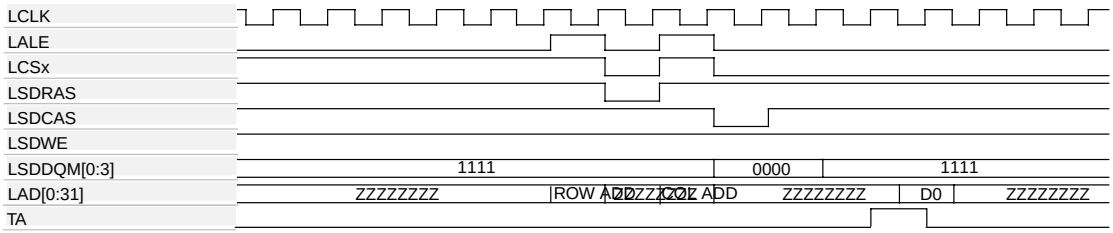


Figure 12-41. SDRAM Single-Beat Read, Page Closed, CL = 3

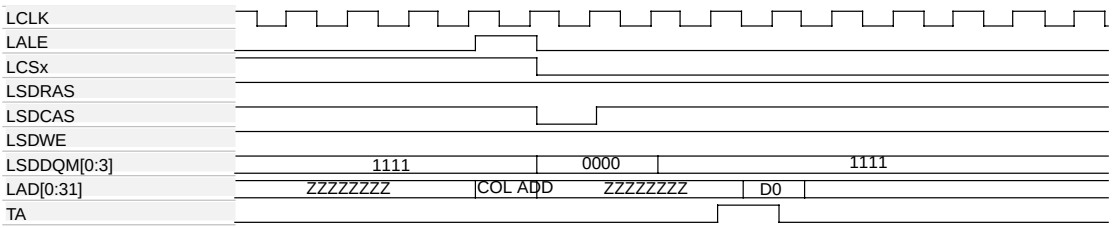


Figure 12-42. SDRAM Single-Beat Read, Page Hit, CL = 3

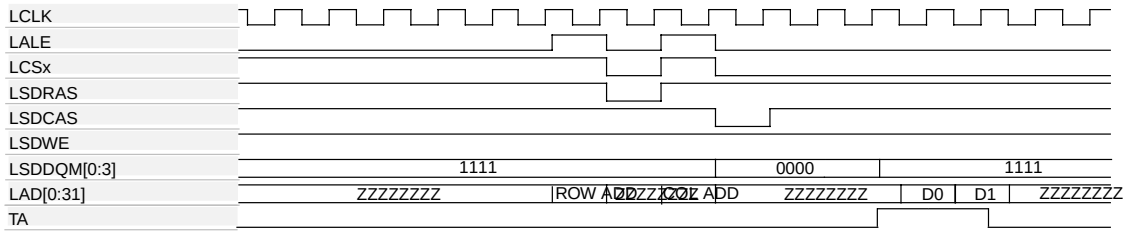


Figure 12-43. SDRAM Two-Beat Burst Read, Page Closed, CL = 3

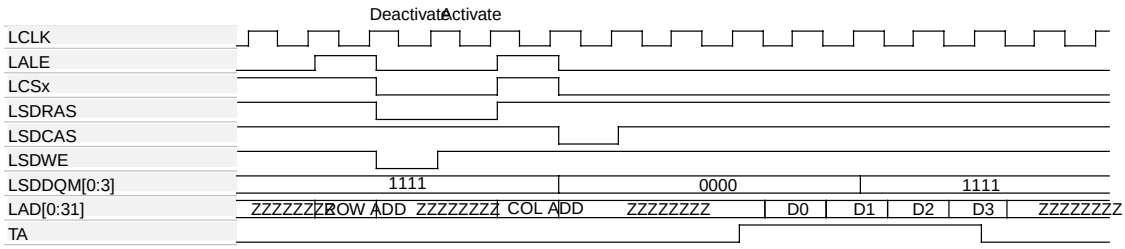


Figure 12-44. SDRAM Four-Beat Burst Read, Page Miss, CL = 3

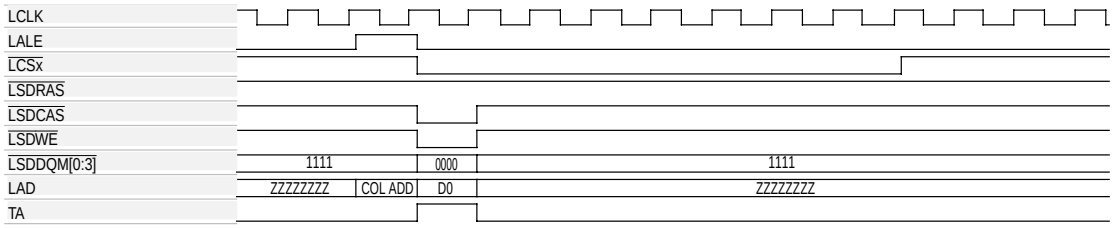


Figure 12-45. SDRAM Single Beat Write, Page Hit.

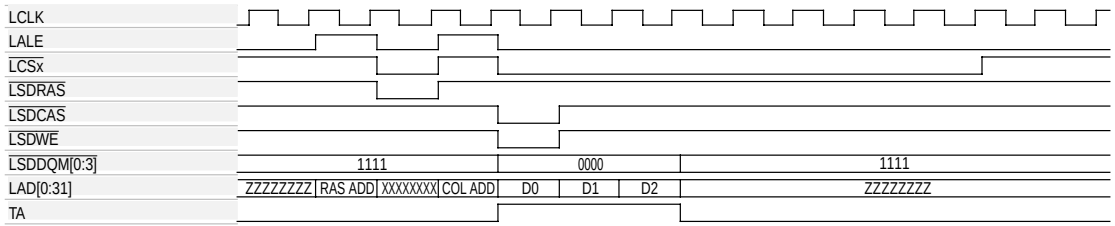


Figure 12-46. SDRAM Three-Beat Write, Page Closed

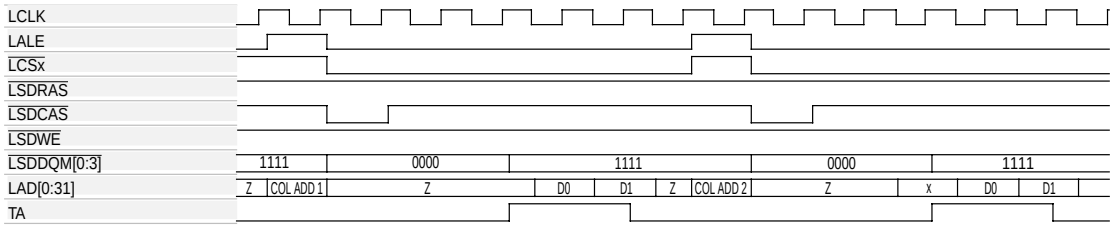


Figure 12-47. SDRAM Read-after-Read Pipelined, Page Hit, CL = 3

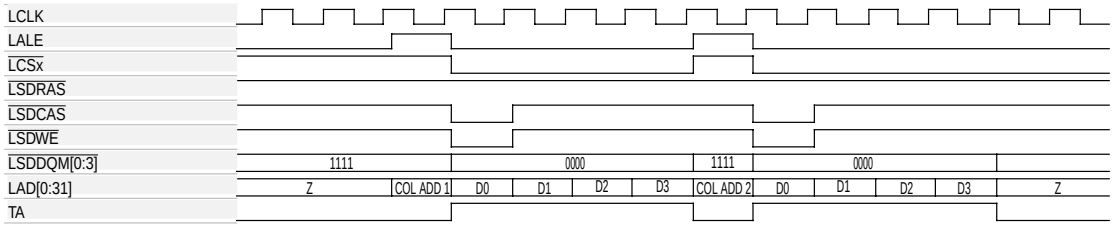


Figure 12-48. SDRAM Write-after-Write Pipelined, Page Hit

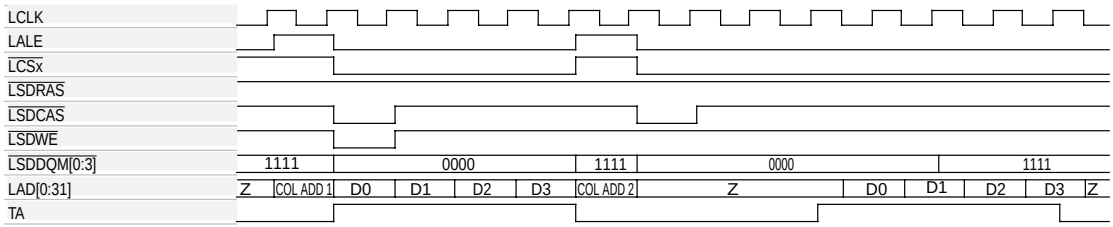


Figure 12-49. SDRAM Read-after-Write Pipelined, Page Hit

12.4.3.9 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.

12.4.3.10 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 12-50. In this case, the latched address carries the mode bits for the command.

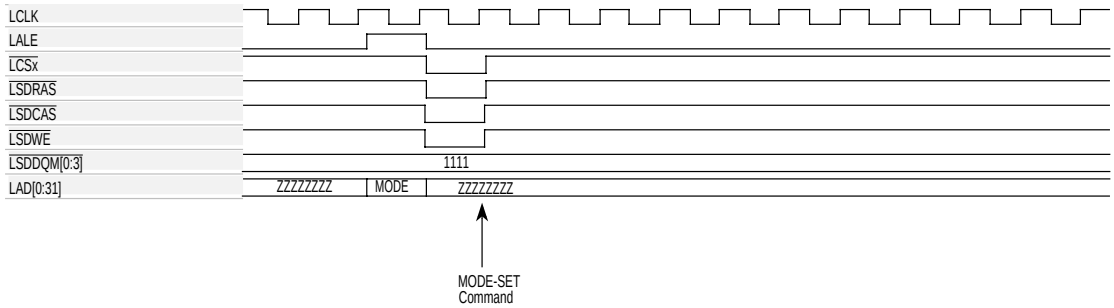


Figure 12-50. SDRAM MODE-SET Command

12.4.3.11 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.

12.4.3.11.1 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

refresh request interval, defined by LSRT and MRTPR, is greater than the refresh recovery interval, defined by LSDMR[RFCR].

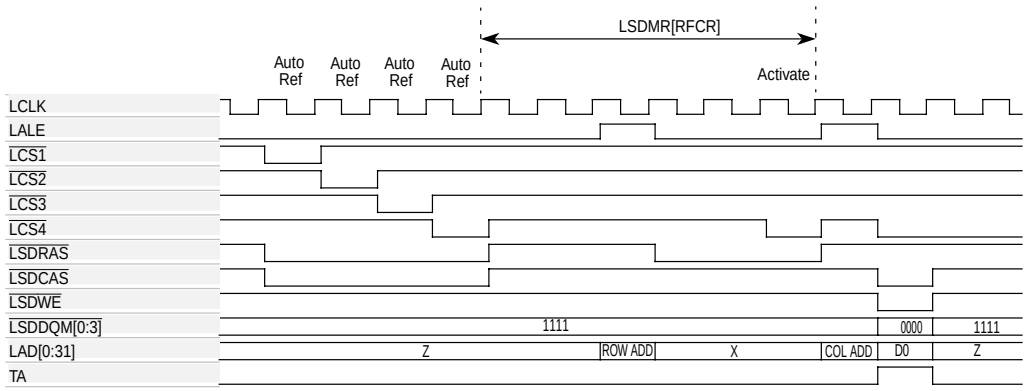


Figure 12-51. SDRAM Bank-Staggered Auto Refresh Timing

12.4.4 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 ($\overline{\text{LCSx}}$, $\overline{\text{LBS}}$[0:3] and $\overline{\text{LGPL}}$[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 12-52 shows the basic operation of each UPM.

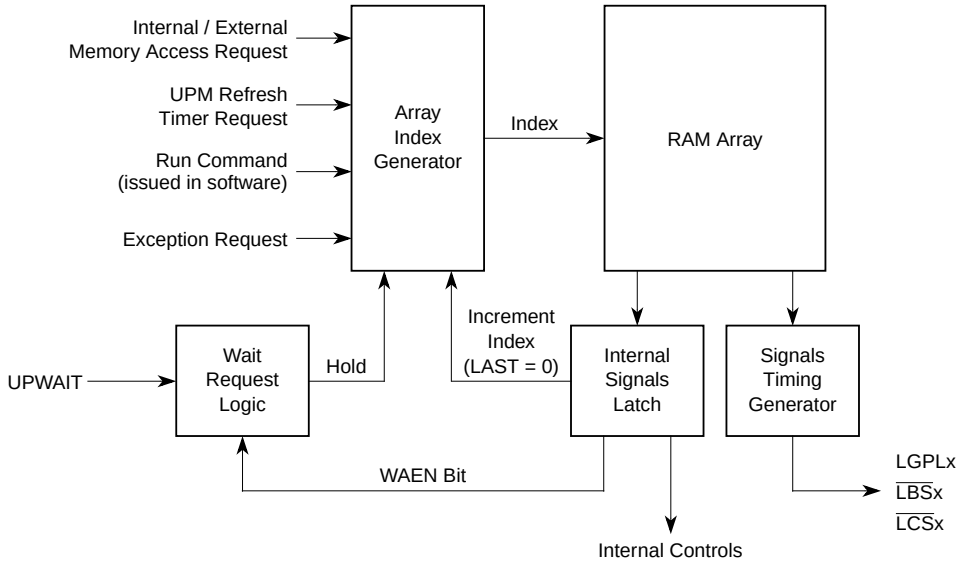


Figure 12-52. User-Programmable Machine Functional Block Diagram

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.

12.4.4.1 UPM Requests

A special pattern location in the RAM array is associated with each of the possible UPM requests. An internal device's request for a memory access initiates one of the following patterns (M*x*MR[OP] = 00):

- Read single-beat pattern (RSS)
- Read burst cycle pattern (RBS)
- Write single-beat pattern (WSS)
- Write burst cycle pattern (WBS)

A UPM refresh timer request pattern initiates a refresh timer pattern (RTS).

An exception (caused by a bus monitor time-out error) occurring while another UPM pattern is running initiates an exception condition pattern (EXS).

Figure 12-53 and Table 12-29 show the start addresses of these patterns in the UPM RAM, according to cycle type. RUN commands (M*x*MR[OP] = 11), however, can initiate patterns starting at any of the 64 UPM RAM words.

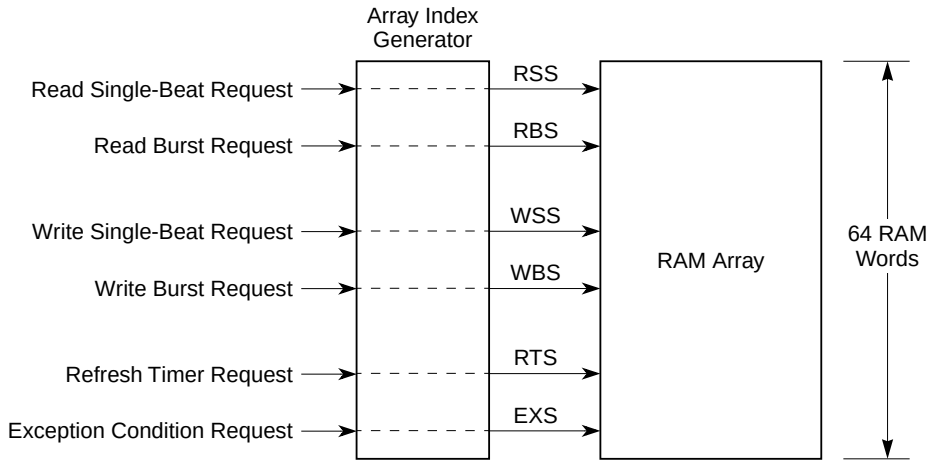


Figure 12-53. RAM Array Indexing

Table 12-29. 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

12.4.4.1.1 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].

12.4.4.1.2 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 12-54 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.

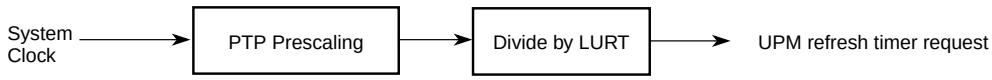


Figure 12-54. Memory Refresh Timer Request Block Diagram

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.

12.4.4.1.3 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 M*x*MR[OP] = 11 and accessing the UPM*x* 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.

12.4.4.1.4 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.

12.4.4.2 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 BR*x* and OR*x* 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 (M*x*MR).

Patterns are written to the RAM array by setting M*x*MR[OP] = 01 and accessing the UPM with any write transaction that hits the relevant chip select. The entire array is thus programmed by an alternating series of writes: to MDR (RAM word to be written) each time followed by a (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).

12.4.4.3 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 12-55 and Figure 12-56. If LCRR[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 LCRR[CLKDIV] = 4 or 8, four phases, T1–T4, define four quarters of the bus clock cycle. Because T2 and T4 are inactive when LCRR[CLKDIV] = 2, UPM ignores signal timing programmed for assertion in either of these phases in the case LCRR[CLKDIV] = 2.

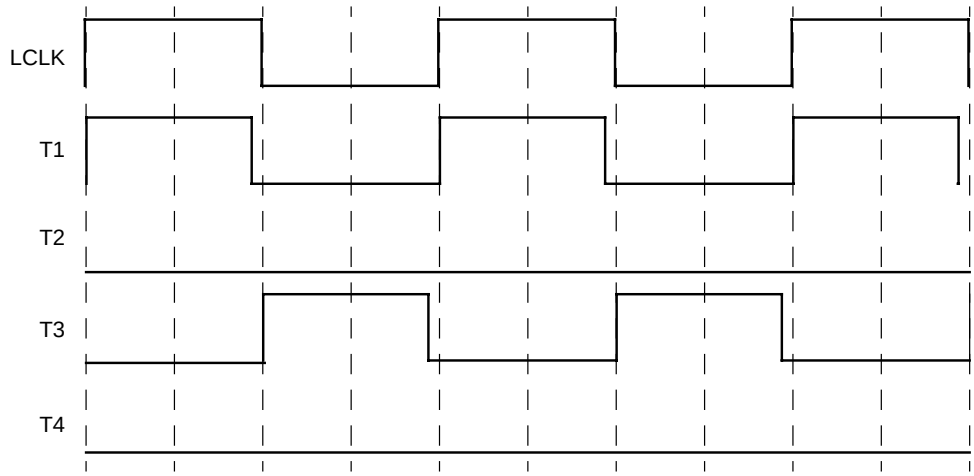


Figure 12-55. UPM Clock Scheme for LCRR[CLKDIV] = 2

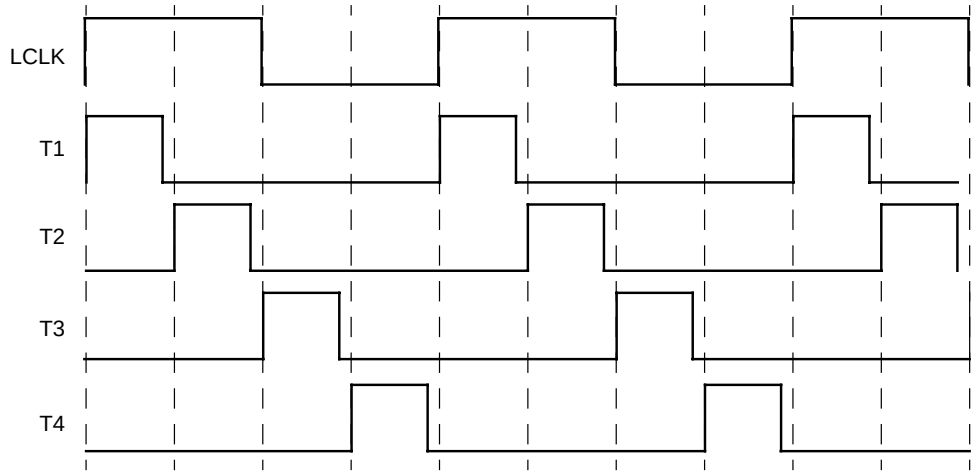


Figure 12-56. UPM Clock Scheme for LCRR[CLKDIV] = 4 or 8

12.4.4.4 The RAM Array

The RAM array for each UPM is 64 locations deep and 32 bits wide, as shown in Figure 12-57. The signals at the bottom of the figure are UPM outputs. The selected $\overline{\text{LCS}x}$

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.

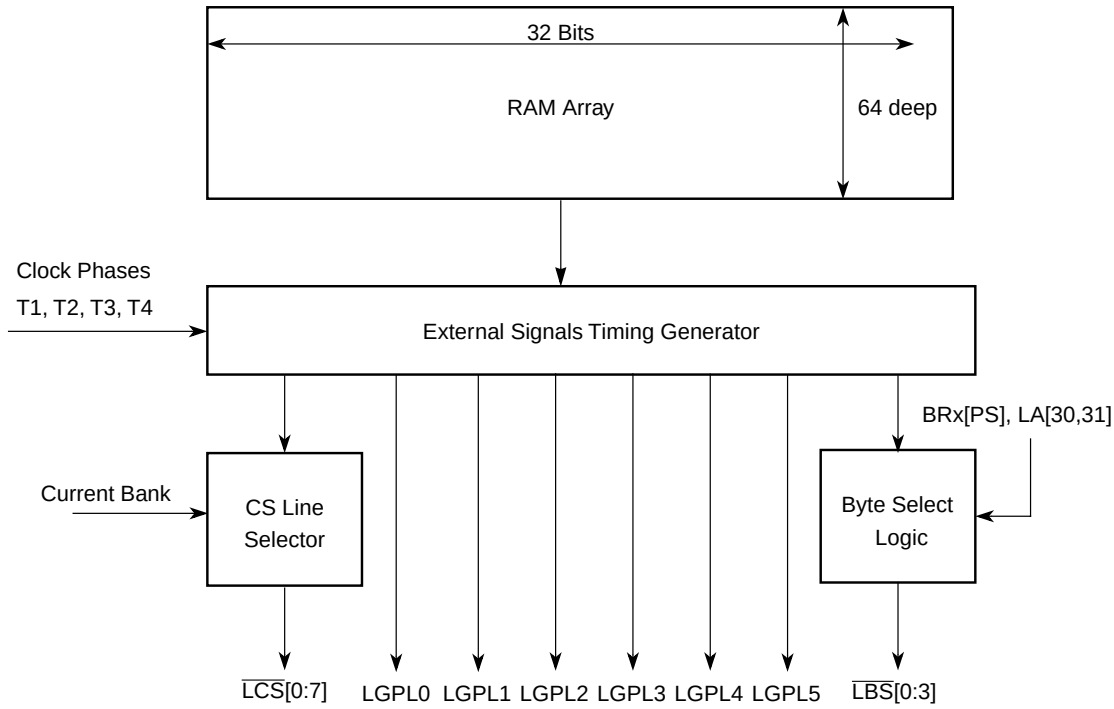


Figure 12-57. RAM Array and Signal Generation

12.4.4.4.1 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 12-58 shows the RAM word fields. When LCRR[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 LCRR[CLKDIV] = 2, CST2 and CST4 are ignored and the external has the values defined by CST1 and CST3 but extended to half the clock cycle in duration. The same interpretation occurs for the BSTx bits when LCRR[CLKDIV] = 2.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CST1	CST2	CST3	CST4	BST1	BST2	BST3	BST4	G0L		G0H		G1T1	G1T3	G2T1	G2T3
Reset	0000_0000_0000_0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	G3T1	G3T3	G4T1 DLT3	G4T3 WAEN	G5T1	G5T3	REDO		LOOP	EXEN	AMX		NA	UTA	TODT	LAST
Reset	0000_0000_0000_0000															
Offset																

Figure 12-58. RAM Word Field Descriptions

Table 12-30 describes RAM word fields.

Table 12-30. 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 LCRR[CLKDIV] = 4 or 8. Defines the state (0 or 1) of LCSx during bus clock half phase 1 if LCRR[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 LCRR[CLKDIV] = 4 or 8. Ignored when LCRR[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 LCRR[CLKDIV] = 4 or 8. Defines the state (0 or 1) of LCSx during bus clock half phase 2 if LCRR[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 LCRR[CLKDIV] = 4 or 8. Ignored when LCRR[CLKDIV] = 2.
4	BST1	Byte select timing1. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 1 (LCRR[CLKDIV] = 4 or 8) or bus clock half phase 1 (LCRR[CLKDIV] = 2), in conjunction with BRx[PS] and 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 (LCRR[CLKDIV] = 4 or 8), in conjunction with BRx[PS] and LA[30-31]. Ignored when LCRR[CLKDIV] = 2.
6	BST3	Byte select timing 3. Defines the state (0 or 1) of $\overline{\text{LBS}}$ during bus clock quarter phase 3 (LCRR[CLKDIV] = 4 or 8) or bus clock half phase 2 (LCRR[CLKDIV] = 2), in conjunction with BRx[PS] and 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 (LCRR[CLKDIV] = 4 or 8), in conjunction with BRx[PS] and LA[30-31]. Ignored when LCRR[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 MxMR[G0CL] 01 reserved 10 0 11 1

Table 12-30. 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 12-30. 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 M*x*MR[AM*x*]. 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 BR*x*[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 Inidicates 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.

12.4.4.4.2 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 12-59 shows how UPMs control $\overline{\text{LCSx}}$ signals.

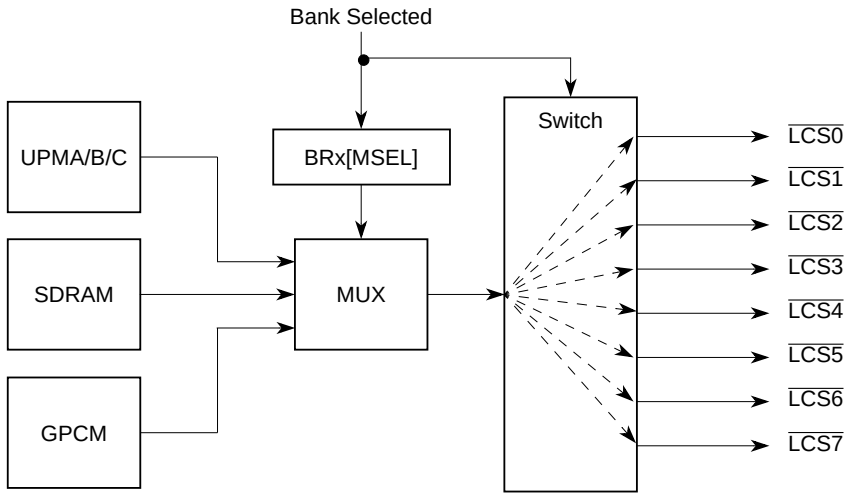


Figure 12-59. $\overline{\text{LCSx}}$ Signal Selection

12.4.4.4.3 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 12-60 shows how UPMs control $\overline{\text{LBS}}$[0:3] signals.

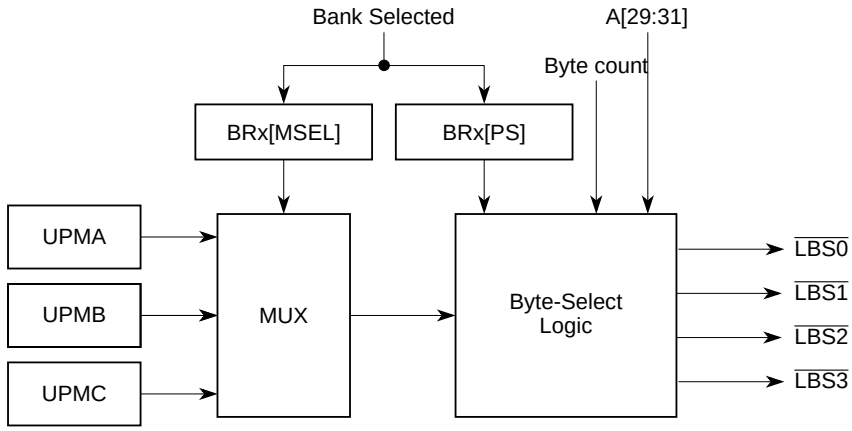


Figure 12-60. $\overline{\text{LBS}}$ Signal Selection

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.

12.4.4.4.4 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.

12.4.4.4.5 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 12-31. 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 12-31. 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

12.4.4.4.6 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.

12.4.4.4.7 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 lsbs 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 12-32. 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.

12.4.4.4.8 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 12-61 shows how data sampling is controlled by the UPM.

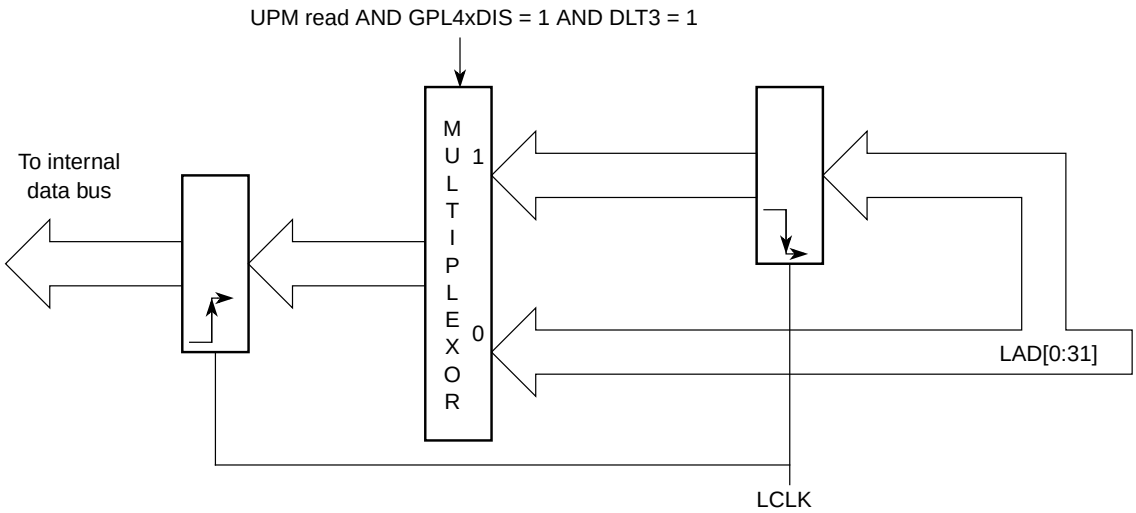


Figure 12-61. UPM Read Access Data Sampling

12.4.4.4.9 $\overline{\text{LGPL}}$[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.

12.4.4.4.10 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 12-62 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

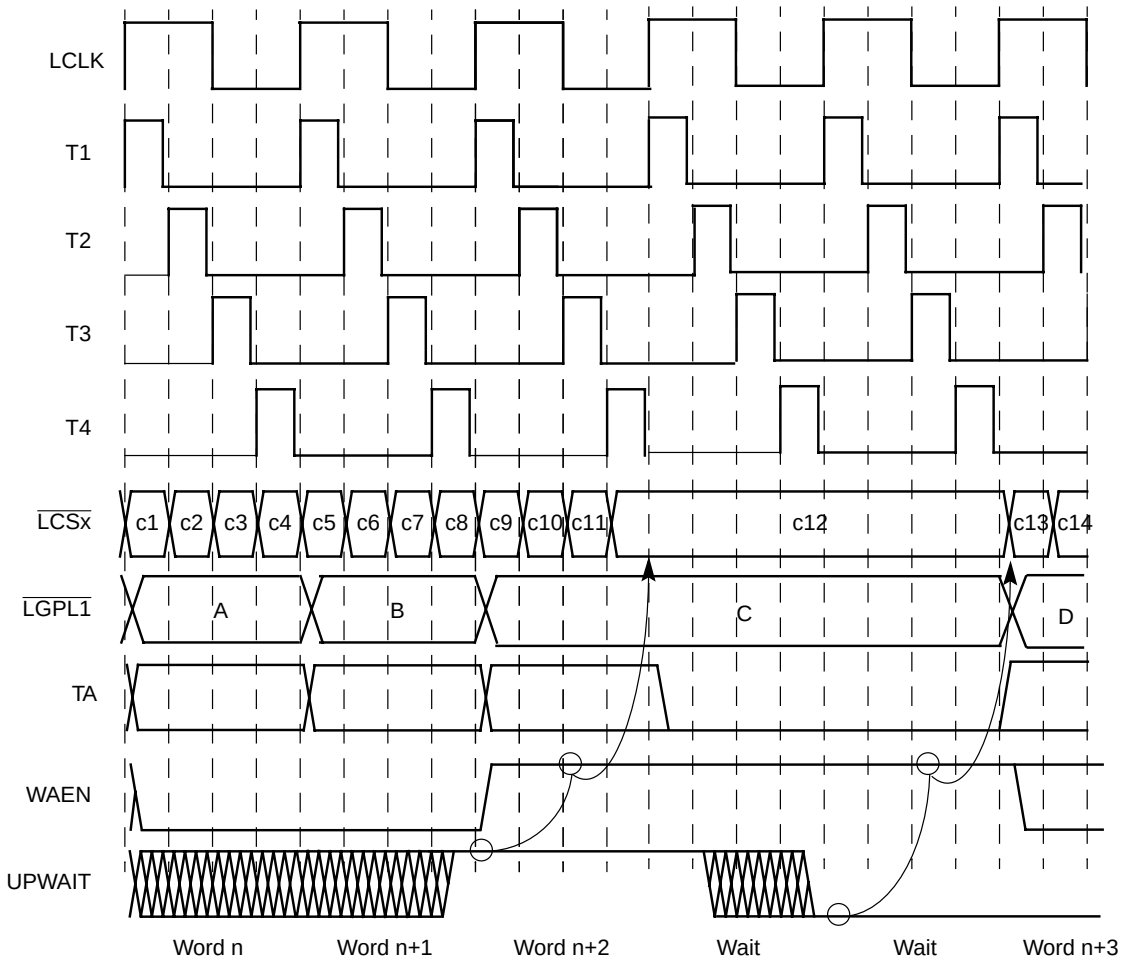


Figure 12-62. Effect of UPWAIT Signal

12.4.4.5 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.

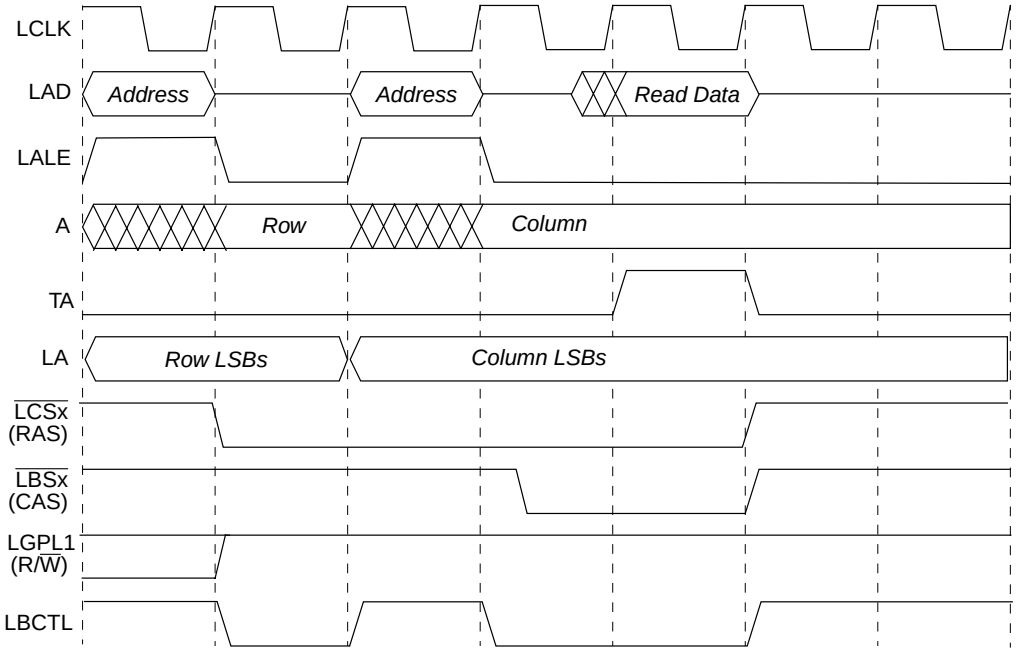
However, programming WAEN = 1 and UTA = 1 in the same RAM word allows UPM to treat UPWAIT as a synchronous signal, which must meet set-up and hold times in relation to the rising edge of the bus clock. In this case, as soon as UPM samples UPWAIT negated on the rising edge of the bus clock, it immediately 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.

12.4.4.6 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.

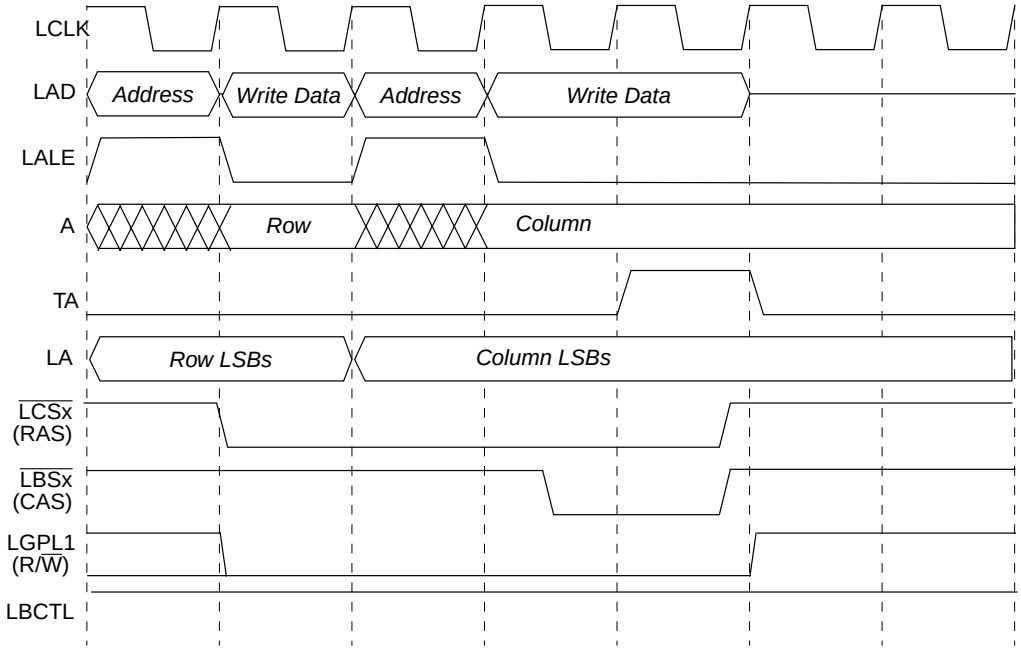
12.4.4.7 Memory System Interface Example Using UPM

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 R/$\overline{W}$ of the DRAM, although any LGPL*x* signal may be used for this purpose.



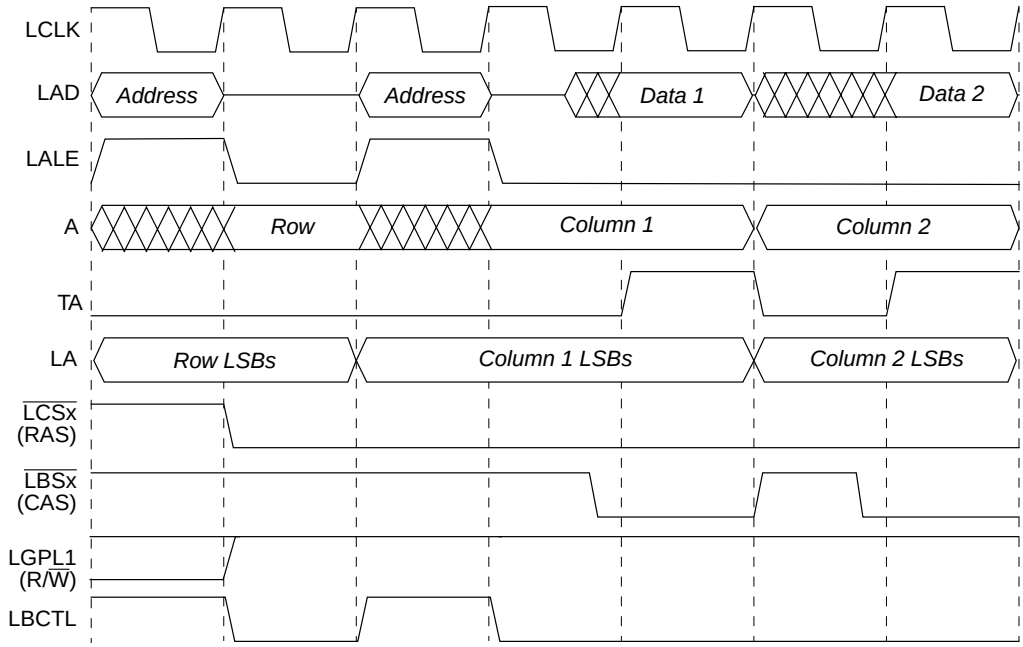
cst1 cst2 cst3 cst4	Bit 0 Bit 1 Bit 2 Bit 3	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 0
bst1 bst2 bst3 bst4	Bit 4 Bit 5 Bit 6 Bit 7	1 1 1 1		1 0 0 0	0 0 0 0
g0l0 g0l1 g0h0 g0h1	Bit 8 Bit 9 Bit 10 Bit 11				
g1t1 g1t3	Bit 12 Bit 13	1 1		1 1	1 1
g2t1 g2t3	Bit 14 Bit 15				
g3t1 g3t3	Bit 16 Bit 17				
g4t1 g4t3	Bit 18 Bit 19				
g5t1 g5t3	Bit 20 Bit 21				
redo[0] redo[1]	Bit 22 Bit 23				
loop exen	Bit 24 Bit 25	0 0		0 0	0 0
amx0 amx1 na uta todt last	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31	1 0 0 0 0 0		0 0 0 0 0 0	0 0 0 1 1 1
	RSS	**RSS**	**RSS+1**	**RSS+1**	**RSS+2**

Figure 12-63. Single-Beat Read Access to FPM DRAM



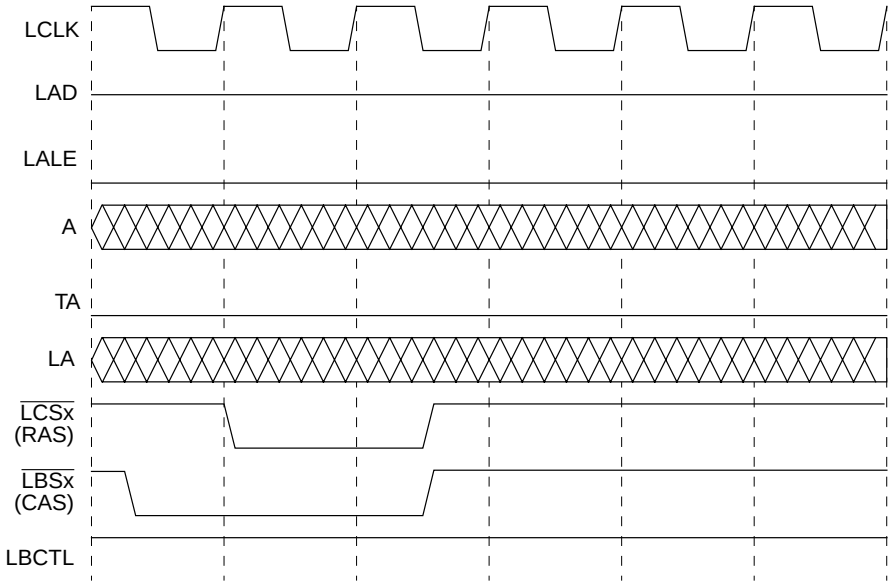
cst1 cst2 cst3 cst4	Bit 0 Bit 1 Bit 2 Bit 3	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 1
bst1 bst2 bst3 bst4	Bit 4 Bit 5 Bit 6 Bit 7	1 1 1 1		1 1 0 0	0 0 0 1
g0l0 g0l1 g0h0 g0h1	Bit 8 Bit 9 Bit 10 Bit 11				
g1t1 g1t3	Bit 12 Bit 13	0 0		0 0	0 0
g2t1 g2t3	Bit 14 Bit 15				
g3t1 g3t3	Bit 16 Bit 17				
g4t1 g4t3	Bit 18 Bit 19				
g5t1 g5t3	Bit 20 Bit 21				
redo[0] redo[1]	Bit 22 Bit 23				
loop exen	Bit 24 Bit 25	0 0		0 0	0 0
amx0 amx1 na uta todt last	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31	1 0 0 0 0 0		0 0 0 0 0 0	0 0 0 1 1 1
	WSS	**WSS**	**WSS+1**	**WSS+1**	**WSS+2**

Figure 12-64. Single-Beat Write Access to FPM DRAM



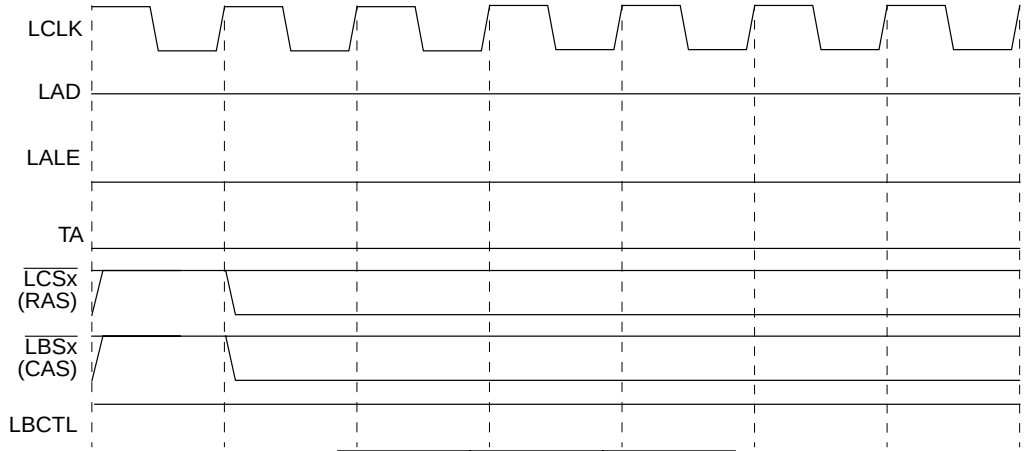
cst1	Bit 0	0	LALE pause (due to change in AMX)	0	0	1
cst2	Bit 1	0		0	0	1
cst3	Bit 2	0		0	0	1
cst4	Bit 3	0		0	0	1
bst1	Bit 4	1		1	0	1
bst2	Bit 5	1		1	0	1
bst3	Bit 6	1		1	0	1
bst4	Bit 7	1		0	0	1
g0l0	Bit 8					
g0l1	Bit 9					
g0h0	Bit 10					
g0h1	Bit 11					
g1t1	Bit 12	1		1	1	1
g1t3	Bit 13	1		1	1	1
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 12-65. Burst Read Access to FPM DRAM Using LOOP (2 Beats Shown)



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 12-66. 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 12-67. Exception Cycle

12.5 Initialization/Application Information

12.5.1 Interfacing to Peripherals

12.5.1.1 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 seperately 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.

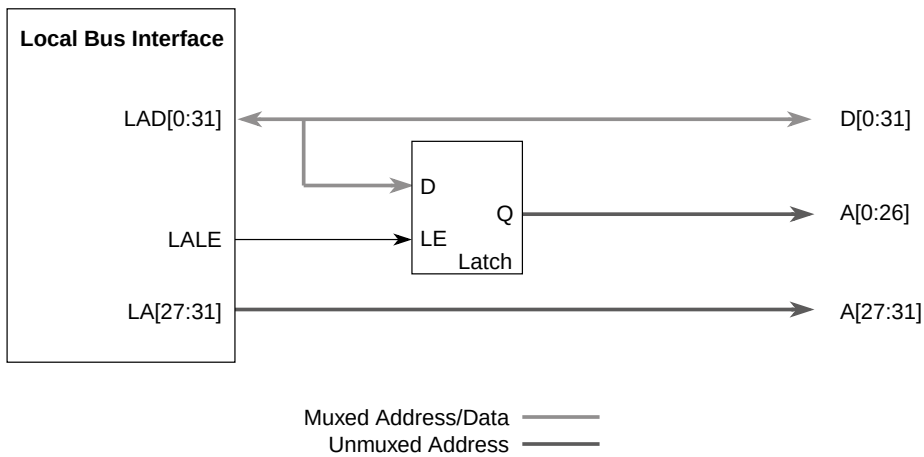


Figure 12-68. Multiplexed Address/Data Bus

12.5.1.2 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 12-69 shows an example of such a hierarchy.

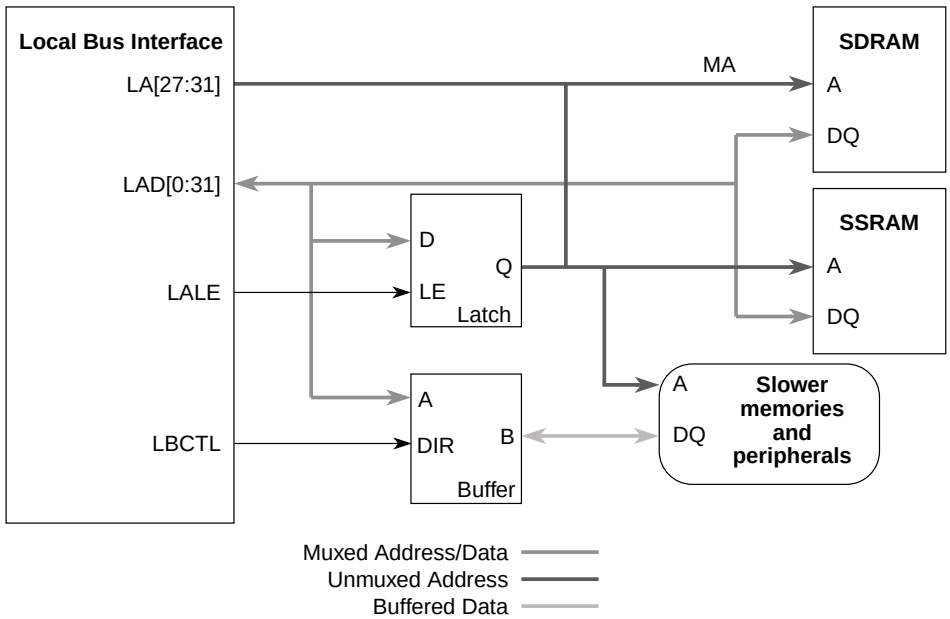


Figure 12-69. Local Bus Peripheral Hierarchy

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.

12.5.1.3 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 futher. For those cases probably only one bank of synchronous SRAMs or SDRAMs should be used and instead of using a seperate latch and a separate bus transceiver, a bus demultiplexor combining those two functions into one device should be used. Figure 12-70 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.

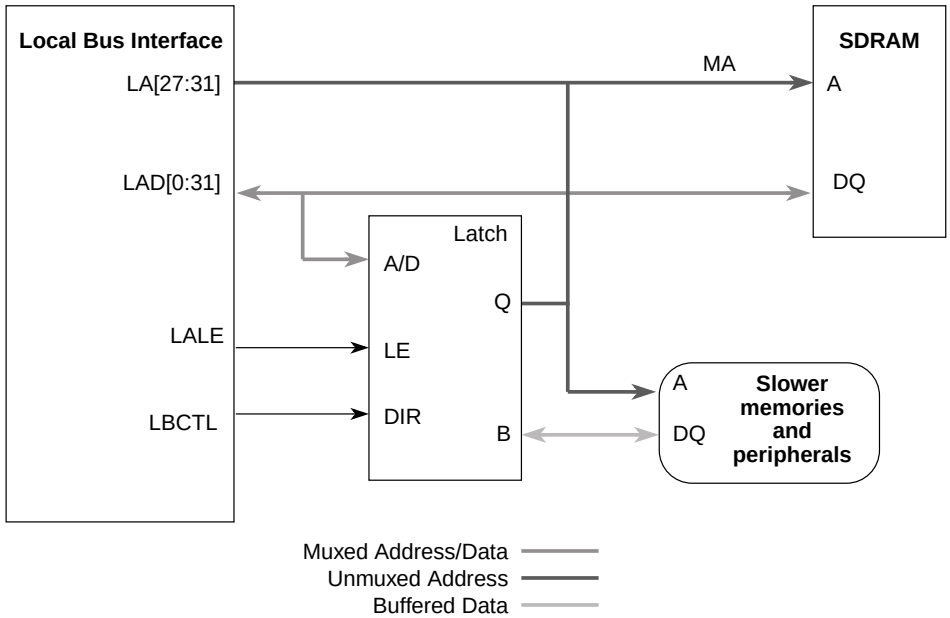


Figure 12-70. Local Bus Peripheral Hierarchy for Very High Bus Speeds

12.5.1.4 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.

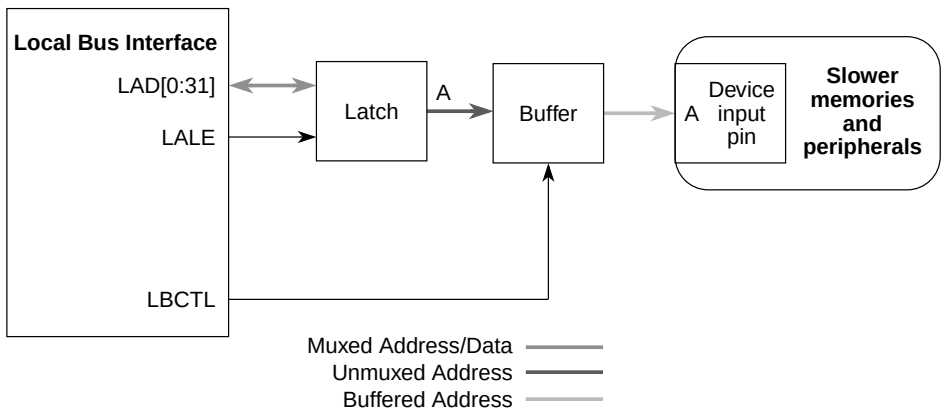


Figure 12-71. GPCM Address Timings

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 $\overline{\text{LCS}}$ 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.

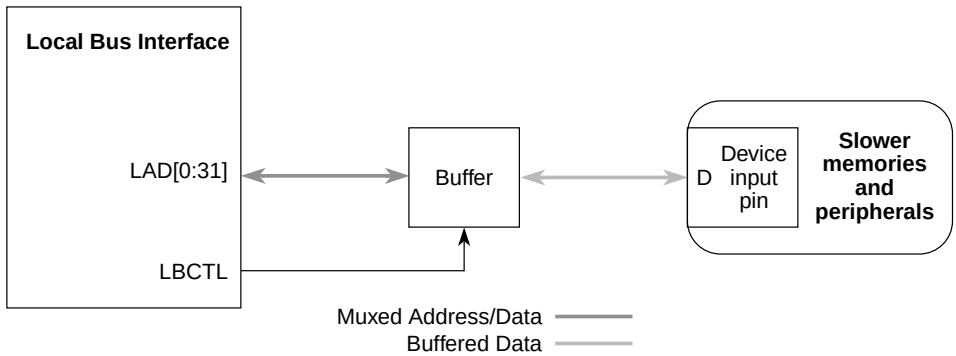


Figure 12-72. GPCM Data timings

12.5.2 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

12.5.2.1 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 impedancee and no bus contention will occur.

12.5.2.2 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}(transceiver) time. The system designer has to ensure, that [t_{en}(LB) + t_{en}(transceiver)] is larger than t_{dis} (LB) to avoid bus contention.

12.5.2.3 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.

12.5.2.4 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.

12.5.3 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 12-73 shows the device connections on the data bus. Table 12-33 lists the bytes required on the data bus for read cycles.

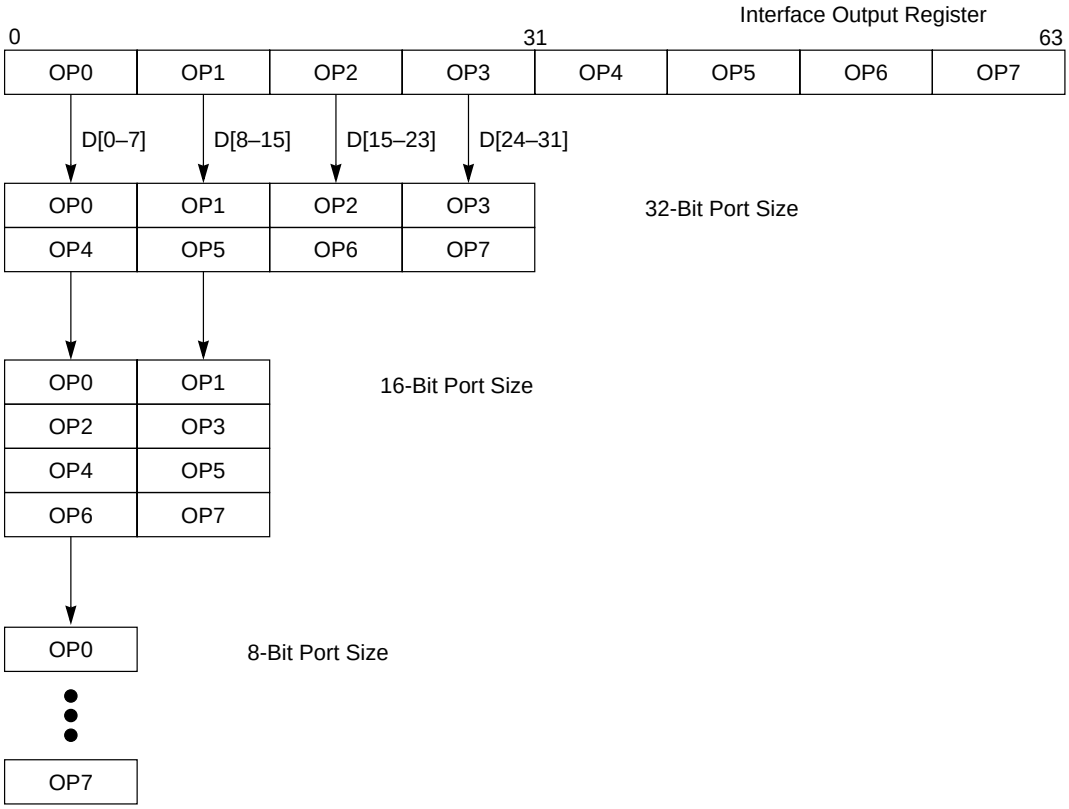


Figure 12-73. Interface to Different Port-Size Devices

Table 12-33. 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 12-33. 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 OP*n*: 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.

12.5.4 Interfacing to SDRAM

12.5.4.1 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 ($\overline{\text{LCS}}$x) 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 (OR*n*) 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:

$$\text{Refresh Period} = \frac{(\text{LSRT} + 1) \times (\text{MPTPR[PTP]} + 1)}{\text{System Frequency}}$$

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 12-51 “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.

12.5.4.2 Maximum Amount of SDRAM Supported

Table 12-34 summarizes information based on SDRAM data sheets supplied by Micron.

Table 12-34. 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.

12.5.4.3 SDRAM Machine Limitations

This section describes limitations of the local bus SDRAM machine.

12.5.4.3.1 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.

12.5.4.3.2 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.

12.5.4.3.3 SDRAM of 128 MBytes

Figure 12-74 shows an SDRAM of 128 Mbytes. Note that all circuit diagrams are principal connection diagrams and do not show any means of signal integrity.

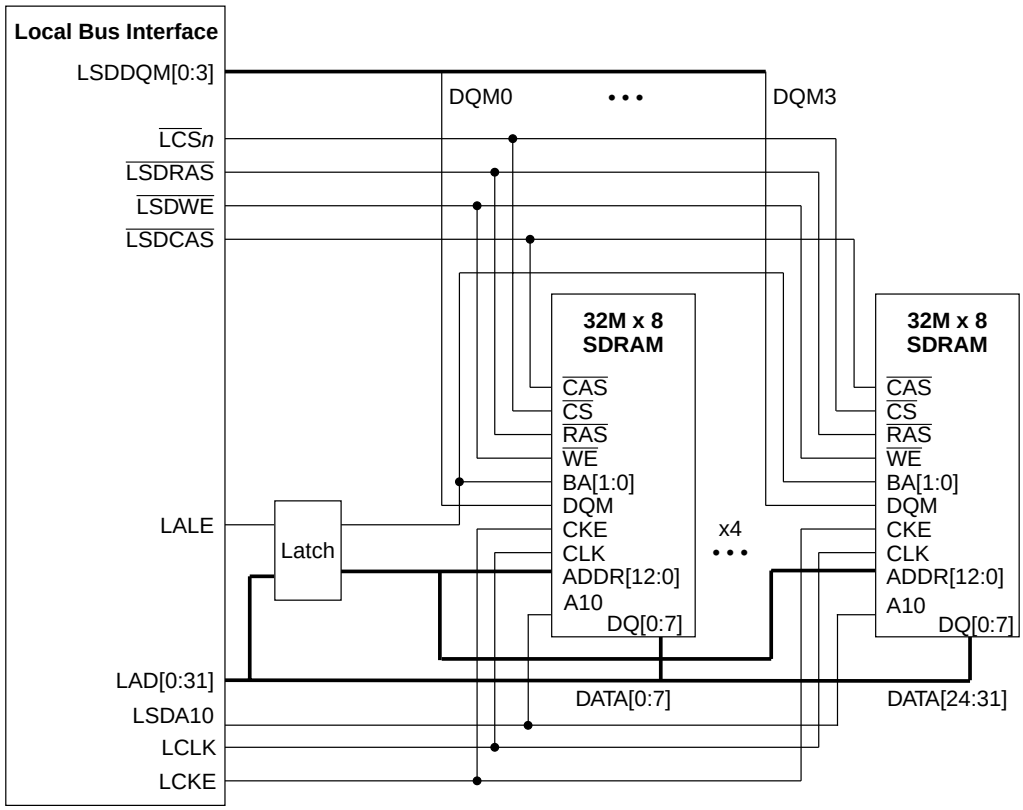


Figure 12-74. 128-Mbyte SDRAM Diagram

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 12-35.

Table 12-35. 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 12-36.

Table 12-36. 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 12-37 shows the address port configuration during a READ/WRITE command.

Table 12-37. 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 12-38 shows the register configuration. PSRT and MPTPR are not shown, but they should be programmed according to the device's specific refresh requirements.

Table 12-38. 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

12.5.4.3.4 256-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.

12.5.4.3.5 512-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 12-39.

Table 12-39. 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 12-40.

Table 12-40. 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 12-41 shows the address port settings during a READ/WRITE command.

Table 12-41. 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 12-42 shows the register configuration. PSRT and MPTPR are not shown, but they should be programmed according to the specific device’s refresh requirements.

Table 12-42. 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

12.5.4.3.6 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 12-75 shows the timing.

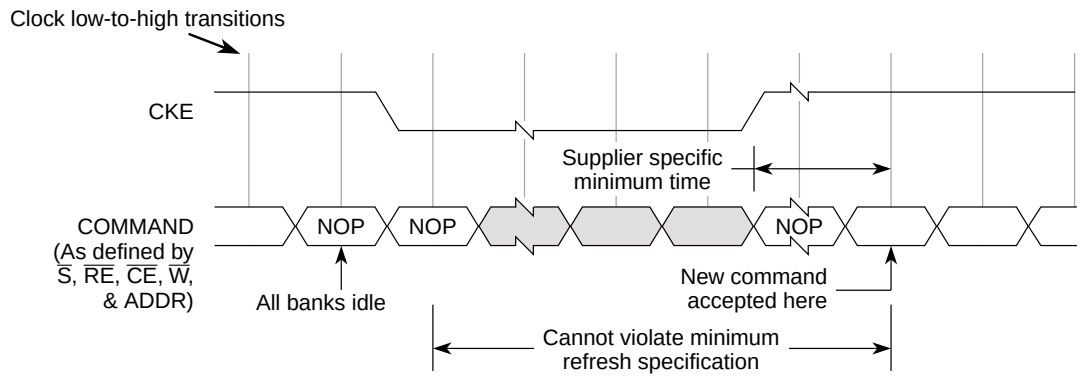


Figure 12-75. SDRAM Power Down Timing

CKE remains low, as long as the device is powered down. After CKE transitions to high, the SDRAM exits the power down mode.

12.5.4.3.7 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 12-76 shows this timing. The SDRAM controller always uses 200 local bus clocks, which should satisfy any SDRAM requirements.

Refer to Section 12.4.3.3, “Intel PC133 and JEDEC-Standard SDRAM Interface Commands,” for SDRAM interface commands and information on the self-refresh command.

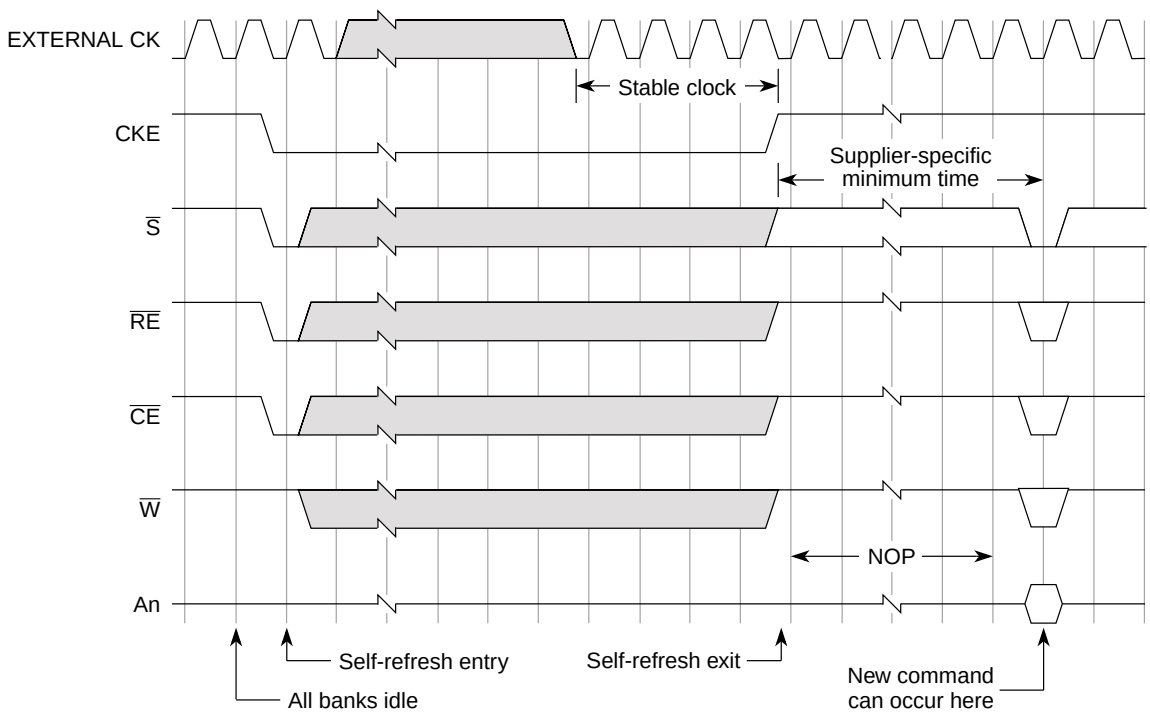


Figure 12-76. SDRAM Self-Refresh Mode Timing

12.5.4.3.8 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 12-43. SDRAM Capacitance

Pin	Min	Max	Unit
CLK	2.0	4.0	pf
$\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$, $\overline{CS}$, 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 12-44. 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 12-77 shows relative timings for local bus clock DLL.

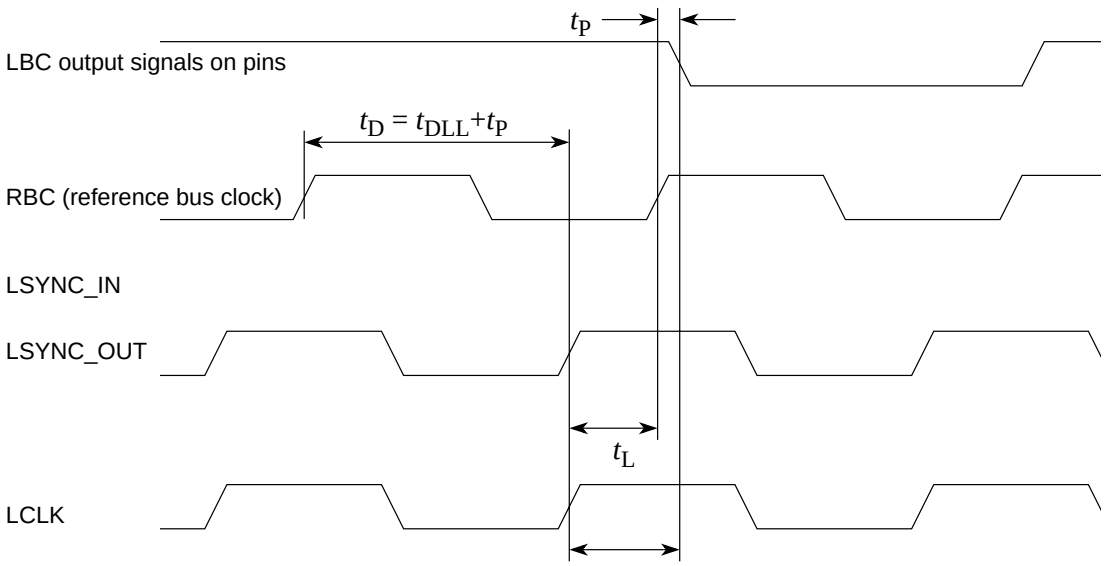


Figure 12-77. Local Bus DLL operation

12.5.4.4 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 12-78 shows a connection diagram.

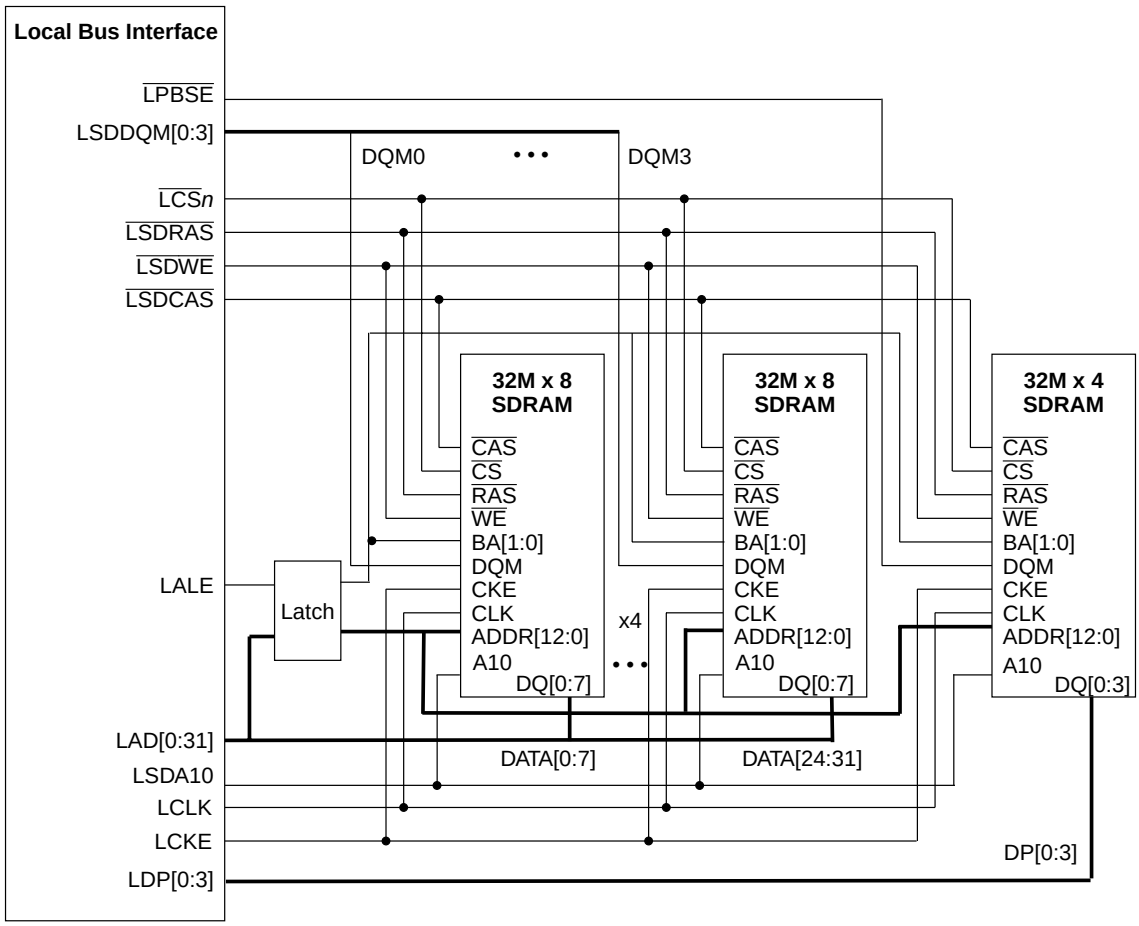


Figure 12-78. Parity Support for SDRAM

12.5.5 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 12-79 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 msut be generated for those cases.

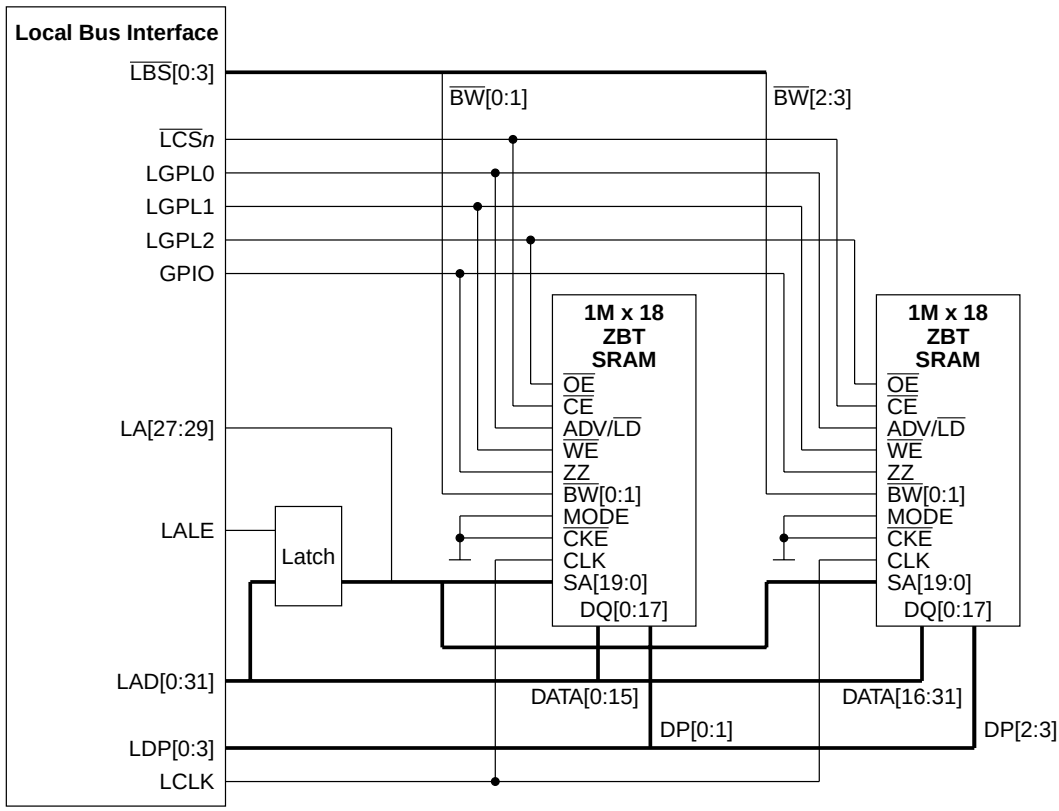


Figure 12-79. Interface to ZBT SRAM

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 $\overline{CKE}$ 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.

12.5.6 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.

12.5.6.1 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.

12.5.6.1.1 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 12-81 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 12.5.6.1.2, "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 12-80 shows the internal register diagram of the HDI16.

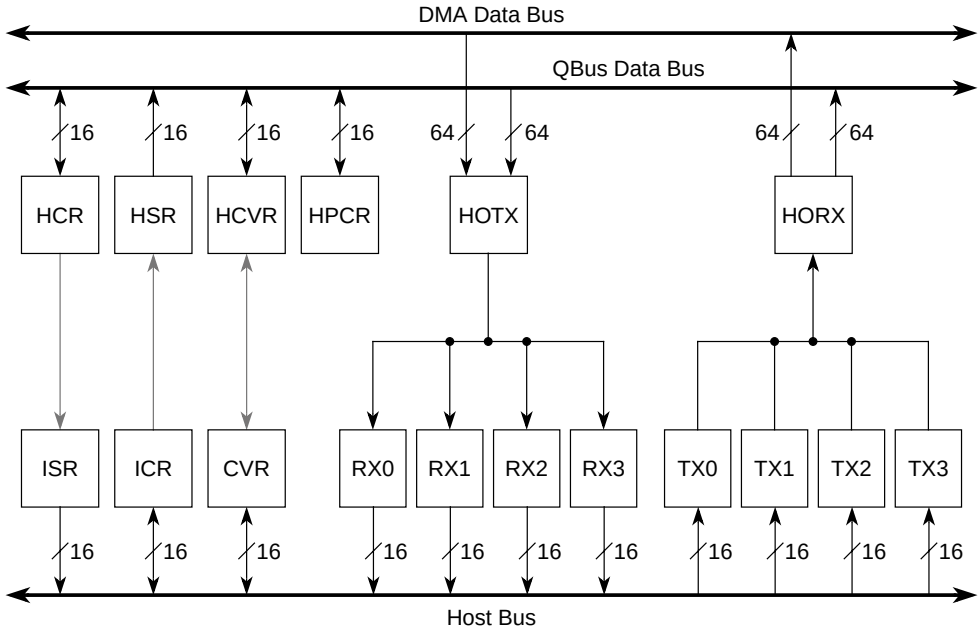


Figure 12-80. MSC8101 HDI16 Peripheral Registers

12.5.6.1.2 Physical Interconnections

The physical interconnections between the UPM controlled local bus and the HDI16 of the MSC8101 HDI16 peripheral are given in Table 12-45 and Figure 12-81.

Table 12-45. 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

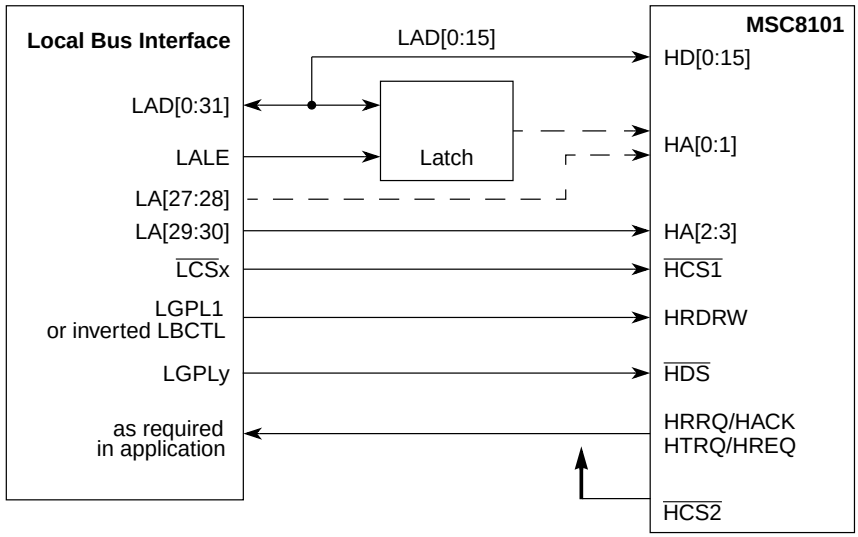


Figure 12-81. Interface to MSC8101 HDI16

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.

12.5.6.1.3 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.

12.5.6.1.4 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.

12.5.6.2 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.

12.5.6.2.1 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 12-82 shows the interface to the MSC8102 DSI for asynchronous mode.

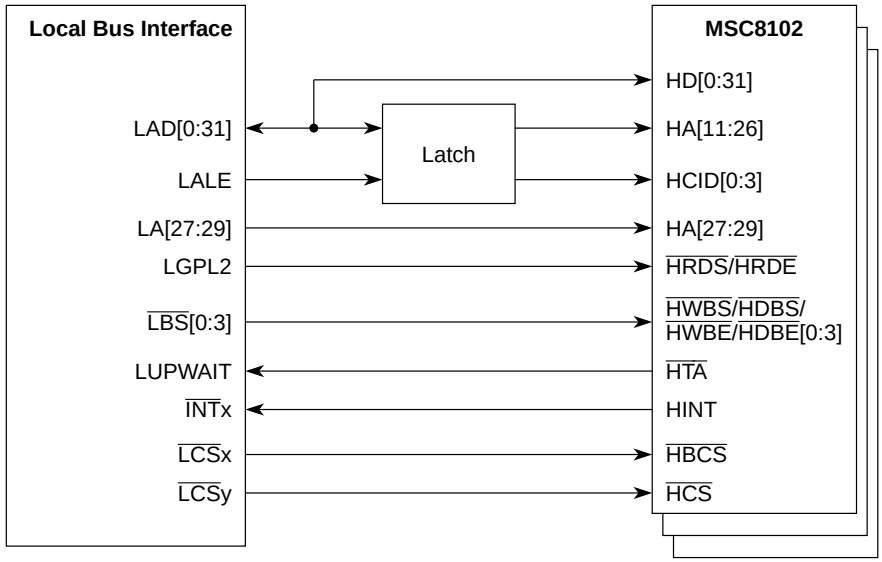


Figure 12-82. Interface to MSC8102 DSI in Asynchronous Mode

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 12-83 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.

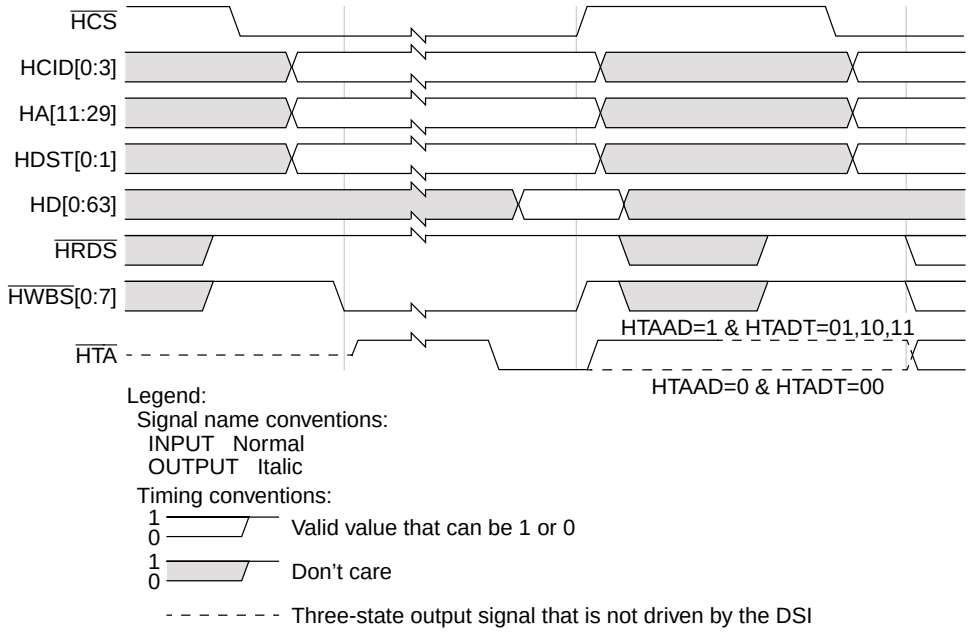


Figure 12-83. Asynchronous Write to MSC8102 DSI

Figure 12-84 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 HTA signal, the host must wait until the previous DSI stops driving the 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 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 HTA will be inactive.

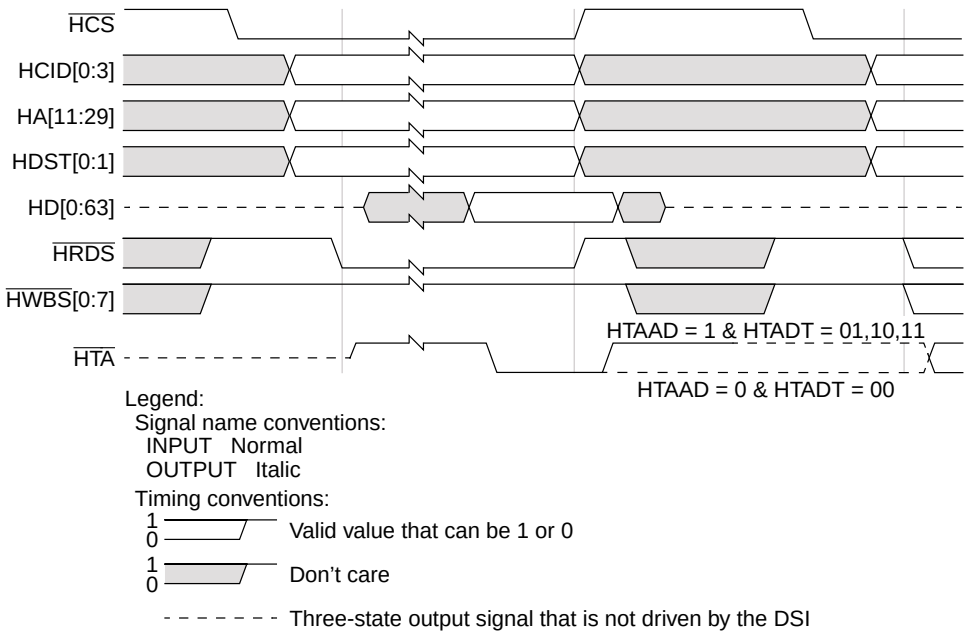


Figure 12-84. Asynchronous Read from MSC8102 DSI

12.5.6.2.2 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 12-85 shows the interface for synchronous mode.

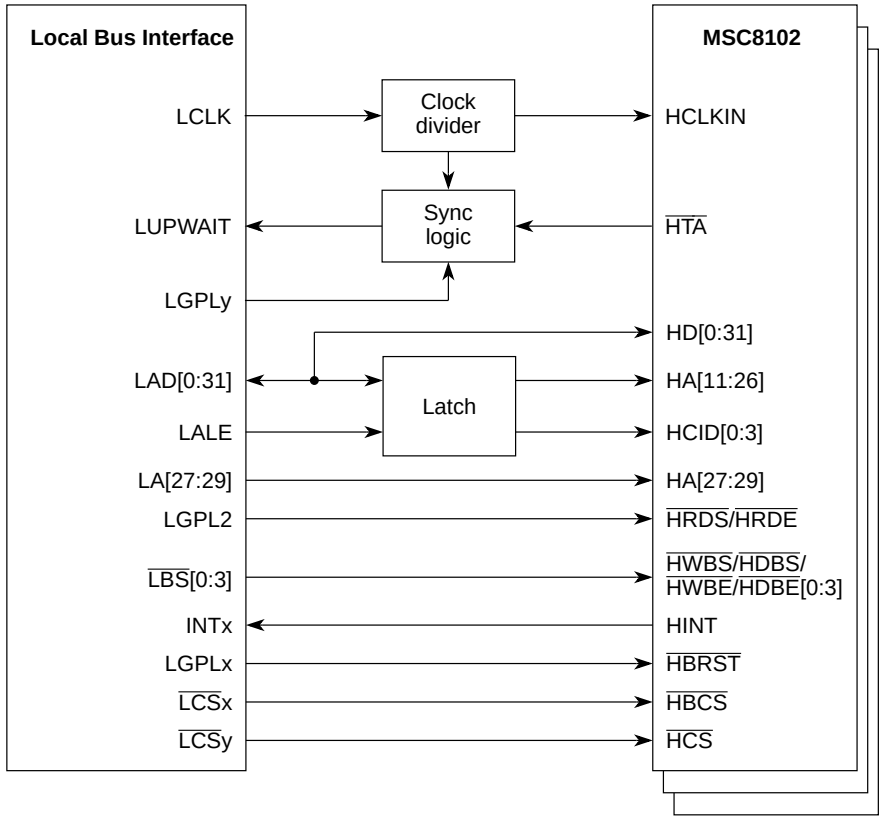


Figure 12-85. Interface to MSC8102 DSI in Synchronous Mode

Because the DSI will assert and negate $\overline{\text{HTA}}$ 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.

$\overline{\text{HTA}}$ 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 $\overline{\text{HTA}}$ 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 $\overline{\text{HTA}}$ signal correctly within one local bus clock. Refer to Section 12.4.4.4.10, “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 devider (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 12.4.4.5, “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 12-86 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.

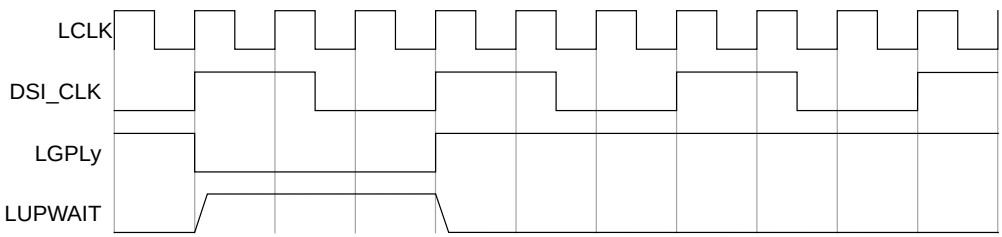


Figure 12-86. UPM Synchronization Cycle

Table 12-46. 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 12-87 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 HCS between accesses. If the next access is not to the same MSC8102, then, to prevent contention on HTA, the host must wait to access the next device until the previous DSI stops driving HTA. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that HTA will be inactive.

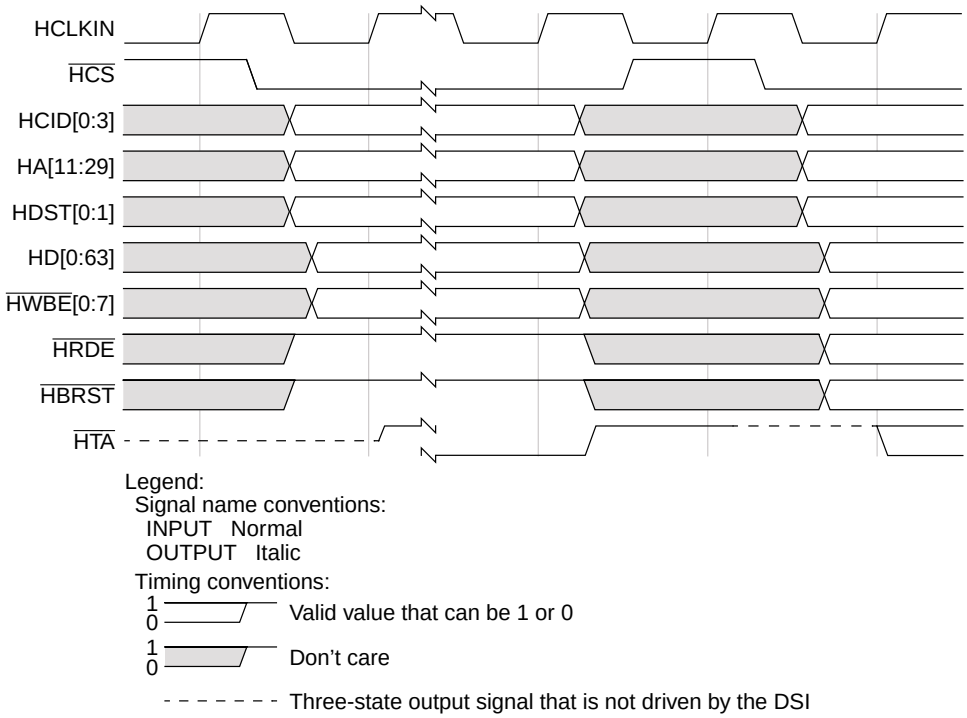


Figure 12-87. Synchronous Single Write to MSC8102 DSI

Synchronous Single Read

Figure 12-88 shows a synchronous single read access. The DSI samples HA, HDST, HCID, HWBE, HRDE, and HBRST on the first HCLKIN rising edge on which HCS is asserted. If the HCID[0:3] signals match the CHIPID value, the DSI is accessed. HRDE is asserted, and HWBE and 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 HTA indicates that data is valid and the host must sample the HD and terminate the access. Because HTA is connected to the UPWAIT signal of the UPM, all local bus signals are frozen until HTA goes to 0 and then the UPM continues in its pattern. The HTA is asserted earlier when the data for this access is already prefetched to the read buffer. 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 HCS between accesses. If

the next access is not to the same MSC8102, then, to prevent contention on HTA, the host must wait to access the next device until the previous DSI stops driving HTA. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that HTA will be inactive.

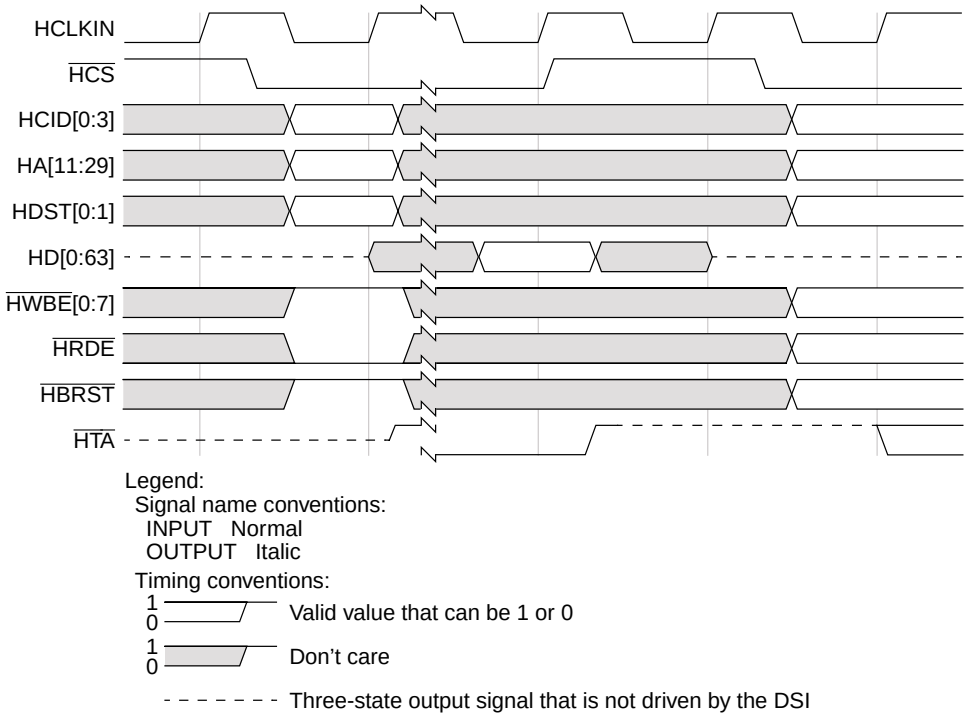


Figure 12-88. Synchronous Single Read to MSC8102 DSI

Synchronous Burst Write

Figure 12-89 shows a synchronous burst write access. The DSI samples HA, HDST, HCID, HD, HWBE, HRDE, and HBRST on the first HCLKIN rising edge on which HCS is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. HWBE are asserted, HBRST is asserted, and HRDE is negated. Assertion of 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 HTA is asserted immediately for each beat of the access. If the write buffer is full, HTA assertion is delayed. Because HTA is connected to the UPWAIT signal of the UPM, all local bus signals are frozen until HTA goes to 0 and then the UPM continues in its pattern. After the last beat of the access, 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 HCS between accesses. If the next access is not to the same MSC8102, then, to prevent contention on HTA, the host must wait to access the next device until the previous DSI stops driving HTA. The easiest

way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that HTA will be inactive.

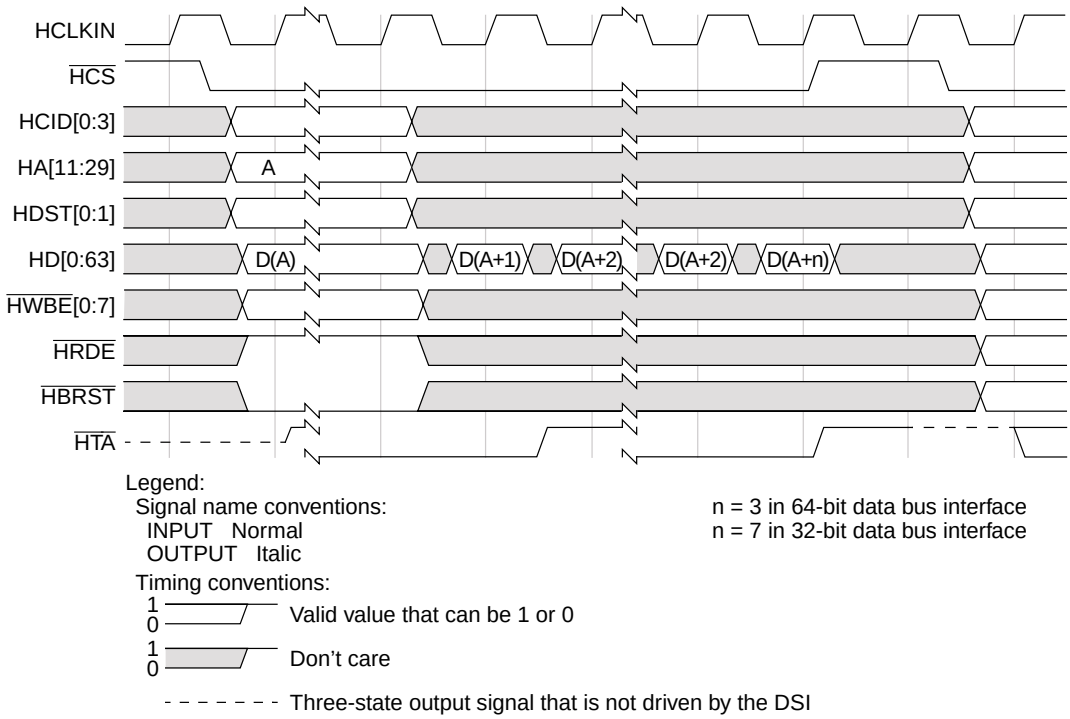


Figure 12-89. Synchronous Burst Write to MSC8102 DSI

Synchronous Burst Read

Figure 12-90 shows a synchronous burst read access. The DSI samples HA, HDST, HCID, HWBE, HRDE, and HBRST on the first HCLKIN rising edge on which HCS is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. HRDE and HBRST are asserted and 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 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 HTA is connected to the UPWAIT signal of the UPM, all local bus signals are frozen until 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 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, HTA remains asserted until the end of the access. After the last beat of the access, HTA is driven to 1 and stops being driven in 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 HCS between accesses. If the next access is not to the same MSC8102, to prevent contention on 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.

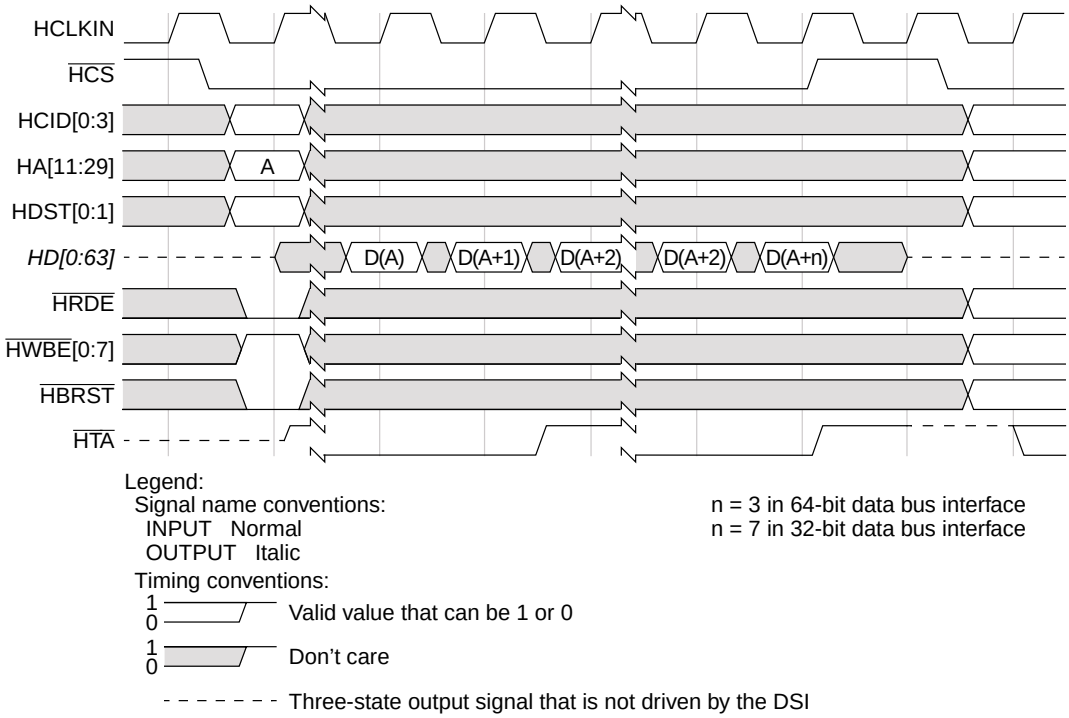


Figure 12-90. Synchronous Burst Read from MSC8102 DSI

12.5.6.2.3 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 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, $\overline{\text{HWBS}}$[0:3]/$\overline{\text{HDBS}}$[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.

12.5.6.3 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 12-47. 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 stobe 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
HCNTL0	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 $\overline{\text{HDS}}$.

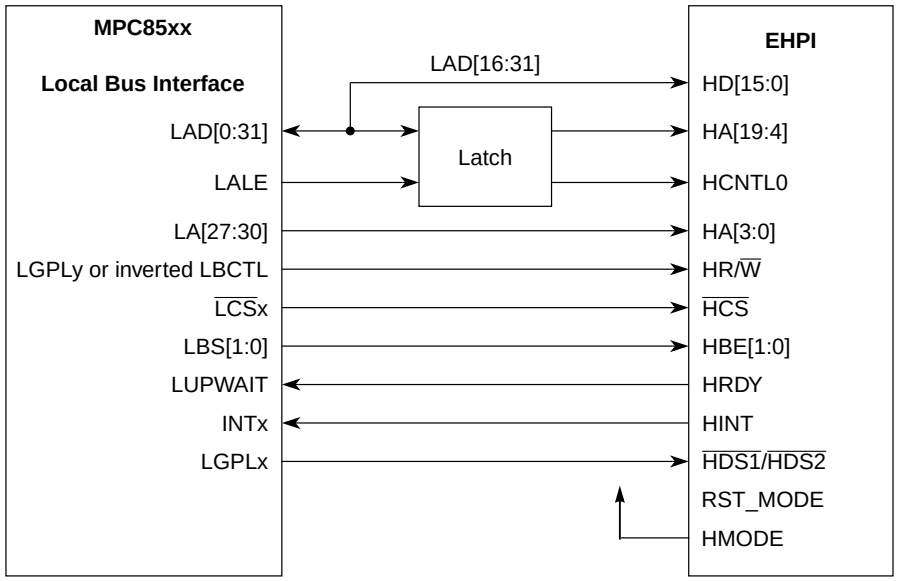
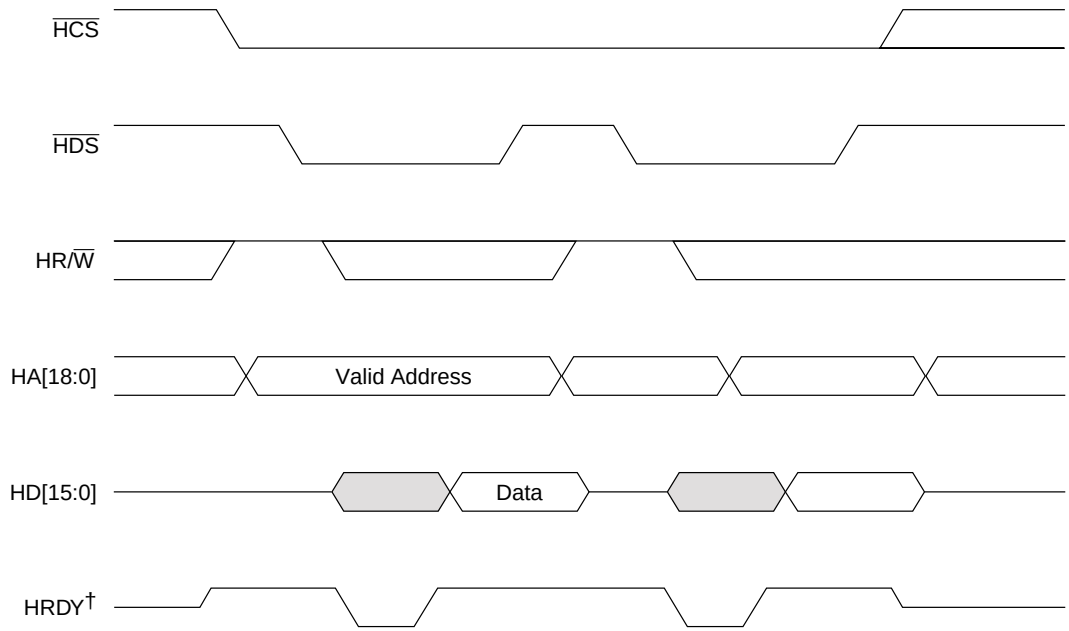


Figure 12-91. Interface to Texas Instruments EHPI in Non-Multiplexed Mode

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 patten 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.

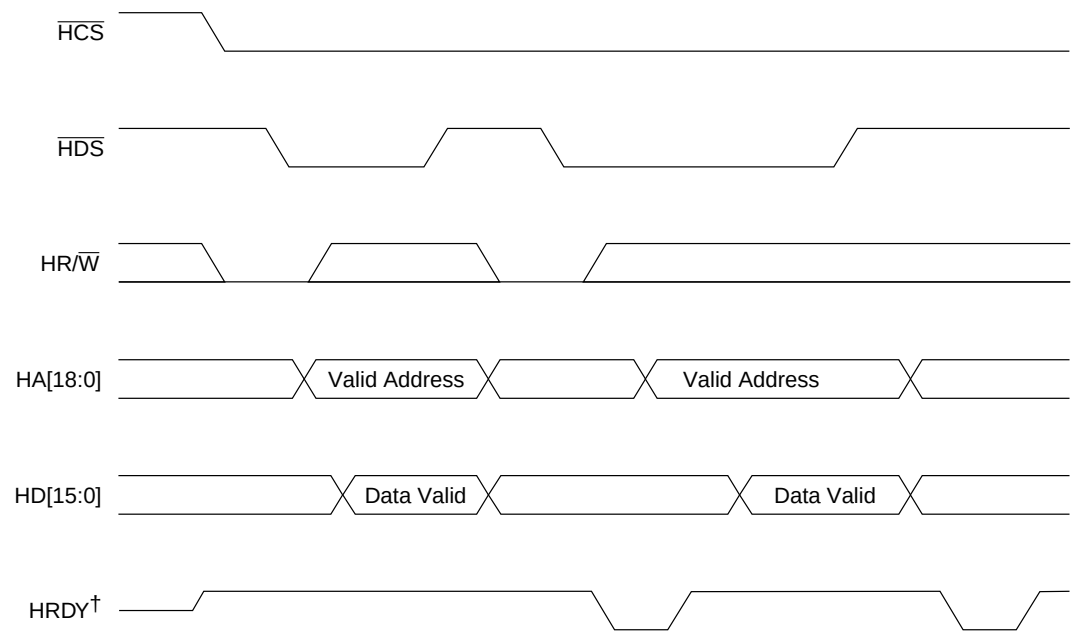
Figure 12-92 shows read timing required by the EHPI in non-multiplexed mode.



† While HCS is not selected, HRDY is in Hi-Z state.

Figure 12-92. EHPI Non-Multiplexed Read Timings

Figure 12-93 shows write timing required by the EHPI in non-multiplexed mode.



† While HCS is not selected, HRDY is in Hi-Z state.

Figure 12-93. EHPI Non-Multiplexed Write Timings

12.5.6.3.1 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 13
Three-Speed Ethernet Controller

13.1 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 13-1 depicts the typical Ethernet protocol stack and the relationship to the OSI model.

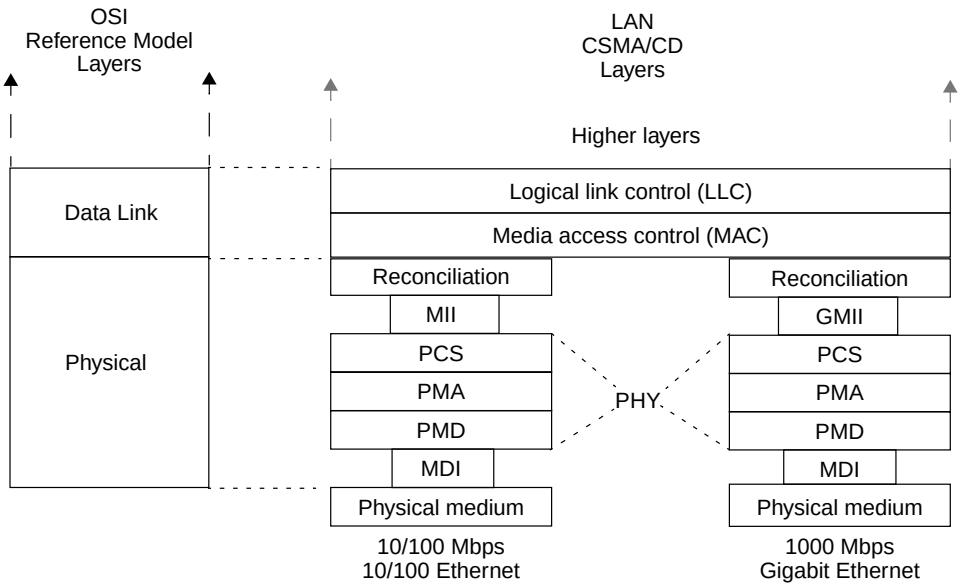


Figure 13-1. Ethernet Protocol in Relation to the OSI Protocol Stack

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 13-2 describes the different physical interface standards.

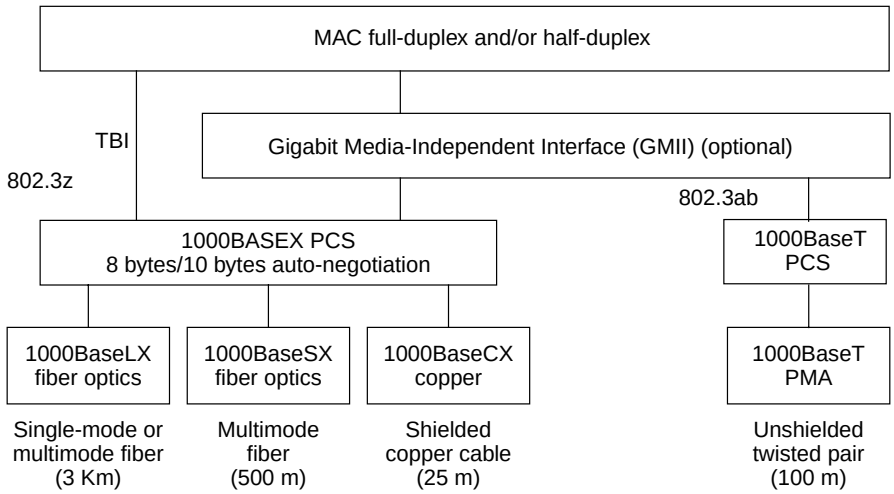


Figure 13-2. 802.3z and 802.3ab Physical Standards

Ethernet/IEEE 802.3 frames are based on the frame structure shown in Figure 13-3. The term “packet” is sometimes used to refer to the frame plus the preamble and start frame delimiter (SFD).

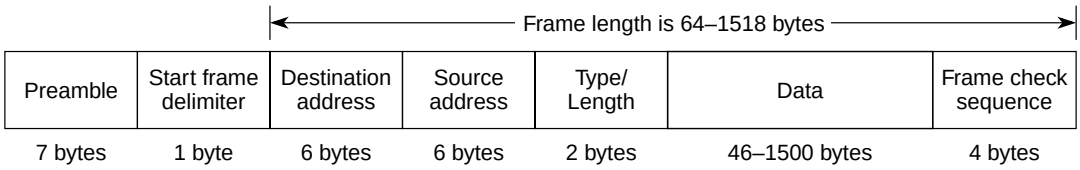


Figure 13-3. Ethernet/IEEE 802.3 Frame Structure

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 13-4 provides additional details of the Ethernet/IEEE 802.3 frame structure.

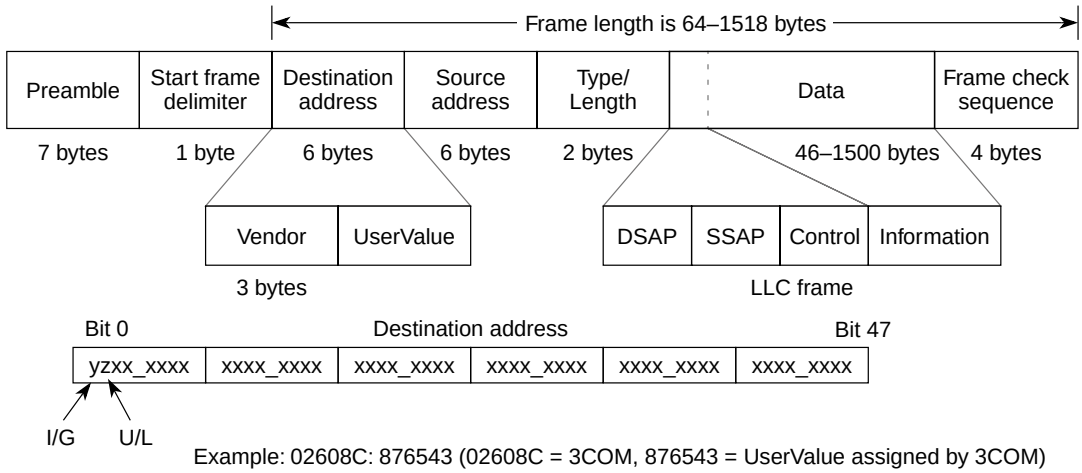


Figure 13-4. Ethernet/IEEE 802.3 Frame Structure With More Details

Relative to Figure 13-4, 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 13-4 (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.

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 13-5 depicts the block diagram of the TSEC.

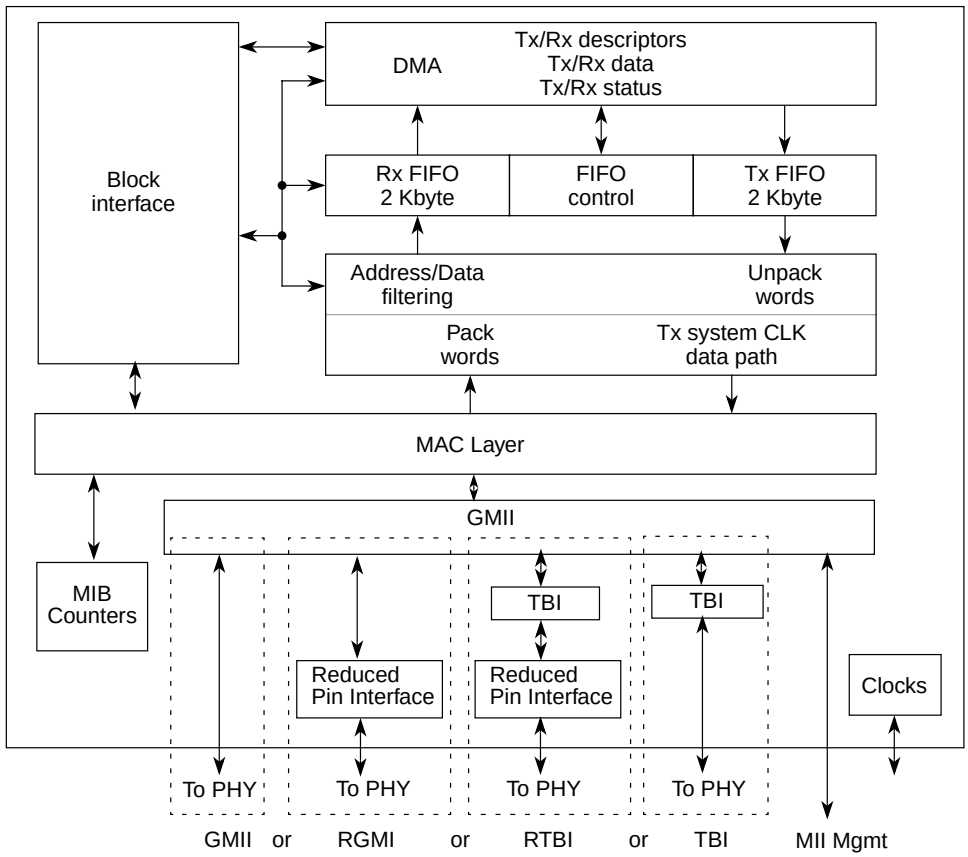


Figure 13-5. TSEC Block Diagram

13.1.1 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.

13.2 Features

The MPC8450 TSEC includes these distinctive features:

- IEEE 802.3, 802.3u, 820.3x, 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

  - 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

13.3 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 13.5.3.8.1, “Individual Address Registers 0–7 (IADDRn),” Section 13.5.3.8.2, “Group Address Registers 0–7 (GADDRn),” Section 13.5.3.4.1, “Receive Control Register (RCTRL),” and Section 13.6.2.6, “Frame Recognition.”

- RMON support

  See Section 13.6.2.5, “RMON Support.”

- Internal loop back

  Internal loop back mode is selected via the loop back bit in the MACCFG1 register. See Section 13.6.2.10, “Internal and External Loop Back,” for details.

13.4 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

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 13-1 lists the network interface signals.

Table 13-1. Signal Properties

Signal Name	Function	Reset State	Pull up
TSEC*n*_COL	MII—collision, input		
TSEC*n*_CRS	MII—carrier sense, input TBI— signal detect, input		
TSEC*n*_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)	
TSEC*n*_RX_CLK	GMII, MII, RGMII—receive clock, input TBI—PMA receive clock 0, input		
TSEC*n*_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		
TSEC*n*_RXD[7:4]	GMII-receive data bit 7:4 input TBI—RCG bit 7:4, input		
TSEC*n*_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		
TSEC*n*_RX_ER	GMII, MII—receive error, input TBI—RGC bit 9, input	0	
TSEC*n*_TX_CLK	MII—transmit clock, input TBI—PMA receive clock 1, input		
TSEC*n*_TXD[7:4]	GMII—transmit data bit 7:4, output TBI—transmit code group (TCG) bit 7:4, output		
TSEC*n*_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 13-1. 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	

13.4.1 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.

13.4.1.1 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.

13.4.1.2 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.

13.4.1.3 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.

13.4.1.4 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.

13.4.1.5 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].

13.4.1.6 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.

13.4.1.7 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.

13.4.1.8 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 represents the 10-bit encoded symbol of GMII receive signals.

13.4.1.9 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 whenTSEC*n*_RX_DV is asserted.

In TBI mode TSEC*n*_RXD[7:4] represents RCG[7:4]. Together, with RCG[9:8] and RCG[3:0], they represent the 10-bit encoded symbol of GMII receive signals.

In GMII mode TSEC*n*_RXD[3:0] represents a nibble of data to be transferred from the PHY to the MAC whenTSEC*n*_RX_DV is asserted. A completely formed SFD must be passed across the MII. While TSEC*n*_RX_DV is not asserted, TSEC*n*_RXD has no meaning.

In RGMII or RTBI mode TSEC*n*_RXD[3:0] are received on the rising edge of TSEC*n*_RX_CLK and TSEC*n*_RXD[7:4] are received on the falling edge of TSEC*n*_RX_CLK.

In TBI mode TSEC*n*_RXD[3:0] represents RCG[3:0]. Together, with RCG[9:4], they represent the 10-bit encoded symbol of GMII receive signals.

13.4.1.10 TSEC*n*_RX_ER—Receive Error (Input)

In GMII or MII mode if TSEC*n*_RX_ER and TSEC*n*_RX_DV are asserted, the PHY has detected an error in the current frame.

In TBI mode TSEC*n*_RX_ER represents RCG[9]. Together, with RCG[8:0], they represent the 10-bit encoded symbol of GMII receive signals.

13.4.1.11 TSEC*n*_TX_CLK—Transmit Clock (Input)

In MII mode TSEC*n*_TX_CLK is a continuous clock (2.5 or 25 MHz) that provides a timing reference for the TSEC*n*_TX_EN, TSEC*n*_TXD, and TSEC*n*_TX_ER signals.

In GMII mode TSEC*n*_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; TSEC*n*_GTX_CLK (125 MHz) becomes the timing reference. The TSEC*n*_GTX_CLK is generated in the TSEC and provided to the PHY and the MAC. The TSEC*n*_TX_CLK is generated in the PHY and provided to the MAC.

In TBI mode this signal is PMA receive clock 1 62.5 MHz, split phase with PMA_RX_CLK1, and is supplied by the SERDES.

13.4.1.12 TSEC*n*_TXD[7:0]—Transmit Data (Output)

In GMII mode TSEC*n*_TXD[7:4], together with TSEC*n*_TXD[3:0], represent one complete octet of data to be sent from the MAC to the PHY when TSEC_TX_DV is asserted and has no meaning while TSEC*n*_TX_EN is negated.

In TBI mode TSEC*n*_TXD[7:4] represents transmit code group (TCG) bits 7:4. Together, with TCG[9:8] and TCG[3:0], they represent the 10-bit encoded symbol.

In GMII or MII mode TSEC*n*_TXD[3:0] represent a nibble of data to be sent from the MAC to the PHY when TSEC*n*_TX_EN is asserted and have no meaning while TSEC*n*_TX_EN is negated.

In RGMII or RTBI mode TSEC*n*_TXD[3:0] are transmitted on the rising edge of TSEC*n*_TX_CLK, and TSEC*n*_TXD[7:4] are transmitted on the falling edge of TSEC*n*_TX_CLK.

In TBI mode TSEC*n*_TXD[3:0] represents TCG[3:0]. Together, with TCG[9:4], they represent the 10-bit encoded symbol.

13.4.1.13 TSEC*n*_TX_EN—Transmit Data Valid (Output)

In GMII or MII mode if TSEC*n*_TX_EN is asserted, the MAC is indicating that valid data is present on the GMII or the MII.

In RGMII mode TSEC*n*_TX_EN becomes TX_CTL. TX_EN and TX_ERR are asserted on this pin on rising and falling edges of the TSEC*n*_TX_CLK, respectively.

In TBI mode TSEC*n*_TX_EN represents TCG[8]. Together, with TCG[9] and TCG[7:0], they represent the 10-bit encoded symbol.

13.4.1.14 TSEC*n*_TX_ER—Transmit Error (Output)

In GMII or MII mode, assertion of TSEC*n*_TX_ER for one or more clock cycles while TSEC*n*_TX_EN is asserted causes the PHY to transmit one or more illegal symbols. Asserting TSEC*n*_TX_ER has no effect while operating at 10Mbps or while TSEC*n*_TX_EN is negated. This signal transitions synchronously with respect to TSEC*n*_TX_CLK.

In TBI mode TSEC*n*_TX_ER represents TCG[9]. Together, with TCG[8:0], they represents the 10-bit encoded symbol.

13.5 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 13.6.3, “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 13.5.4, “Ten-Bit Interface (TBI).”

13.5.1 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 13-2 defines the top-level memory map.

Table 13-2. 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

13.5.2 Detailed Memory Map—Control/Status Registers

Table 13-2 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 13-5, but the registers for TSEC2 are not. Note in Table 13-5 that the registers are the same for TSEC2 except that the offset changes from 0x2_4nnn to 0x2_5nnn.

Table 13-3. 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	13.5.3.1.1/13-21
0x2_4014	Interrupt mask register (IMASK)	R/W	0x0000_0000	13.5.3.1.2/13-24
0x2_4018	Error Disabled register (EDIS)	R/W	0x0000_0000	13.5.3.1.3/13-25
0x2_401C	Reserved	R	0x0000_0000	—
0x2_4020	Ethernet control register (ECNTRL)	R/W	0x0000_0000	13.5.3.1.4/13-26
0x2_4024	Minimum frame length register (MINFLR)	R/W	0x0000_0040	13.5.3.1.5/13-28
0x2_4028	Pause time value register (PTV)	R/W	0x0000_0000	13.5.3.1.6/13-28
0x2_402C	DMA control register (DMACTRL)	R/W	0x0000_0000	13.5.3.1.7/13-29
0x2_4030	TBI PHY address register (TBIPA)	R/W	0x0000_0000	13.5.3.1.8/13-31
0x2_4034–0x2_403C	Reserved	R	0x0000_0000	—
0x2_4040–0x2_4088	Reserved	R	0x0000_0000	—
TSEC FIFO Control and Status Registers				13.5.3.2/13-31
0x2_408C	FIFO transmit threshold register (FIFO_TX_THR)	R/W	0x0000_0100	13.5.3.2.1/13-32
0x2_4090–0x2_4094	Reserved	R	0x0000_0000	—
0x2_4098	FIFO transmit starve register (FIFO_TX_STARVE)	R/W	0x0000_0080	13.5.3.2.2/13-33
0x2_409C	FIFO transmit starve shutoff register (FIFO_TX_STARVE_SHUTOFF)	R/W	0x0000_0100	13.5.3.2.3/13-33
0x2_40A0–0x2_40FC	Reserved	R	0x0000_0000	—
TSEC1 Transmit Control and Status Registers				
0x2_4100	Transmit control register (TCTRL)	R/W	0x0000_0000	13.5.3.3.1/13-34
0x2_4104	Transmit status register (TSTAT)	R/W	0x0000_0000	13.5.3.3.2/13-35
0x2_4108	Reserved	R	0x0000_0000	—
0x2_410C	TxBD data length (TBDLEN)	R/W	0x0000_0000	13.5.3.3.3/13-36
0x2_4110–0x2_4120	Reserved	R	0x0000_0000	—
0x2_4124	Current TxBD pointer (CTBPTR)	R/W	0x0000_0000	13.5.3.3.4/13-36
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 13-3. 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	13.5.3.3.5/13-37
0x2_4208–0x2_42AC	Reserved	R	0x0000_0000	—
0x2_42B0	Out-of-sequence TxBD register (OSTBD)	R/W	0x0000_0000	13.5.3.3.6/13-38
0x2_42B4	Out-of-sequence Tx data buffer pointer register (OSTBDP)	R/W	0x0000_0000	13.5.3.3.7/13-40
0x2_42B8–0x2_42FC	Reserved	R	0x0000_0000	—
TSEC1 Receive Control and Status Registers				
0x2_4300	Receive control register (RCTRL)	R/W	0x0000_0000	13.5.3.4.1/13-41
0x2_4304	Receive status register (RSTAT)	R/W	0x0000_0000	13.5.3.4.2/13-41
0x2_4308	Reserved	R	0x0000_0000	—
0x2_430C	RxBD data length (RBDLEN)	R/W	0x0000_0000	13.5.3.4.3/13-42
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	13.5.3.4.4/13-43
0x2_4328–0x2_433C	Reserved	R	0x0000_0000	—
0x2_4340	Maximum receive buffer length register (MRBLR)	R/W	0x0000_0000	13.5.3.4.5/13-43
0x2_4344–0x2_4380	Reserved	R	0x0000_0000	—
0x2_4384	RxBD pointer (RBPTR)	R/W	0x0000_0000	13.5.3.4.6/13-44
0x2_4388–0x2_4400	Reserved	R/W	0x0000_0000	—
0x2_4404	RxBD base address (RBASE)	R/W	0x0000_0000	13.5.3.4.7/13-45
0x2_4408–0x2_44FC	Reserved	R/W	0x0000_0000	—
TSEC1 MAC Registers				
0x2_4500	MAC configuration #1 (MACCFG1)	R/W, R	0x0000_0000	13.5.3.6.1/13-48
0x2_4504	MAC configuration #2 (MACCFG2)	R/W	0x0000_7000	13.5.3.6.2/13-50
0x2_4508	IPGIFGI	R/W	0x4060_5060	13.5.3.6.3/13-51
0x2_450C	Half-duplex (HAFDUP)	R/W	0x00A1_F037	13.5.3.6.4/13-52
0x2_4510	Maximum frame (MAXFRM)	R/W	0x0000_0600	13.5.3.6.5/13-54
0x2_4514	Reserved	R	0x0000_0000	—

Table 13-3. 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	13.5.3.6.6/13-54
0x2_4524	MII Mgmt: command (MIICOM)	R/W	0x0000_0000	13.5.3.6.7/13-55
0x2_4528	MII Mgmt: address (MIIMADD)	R/W	0x0000_0000	13.5.3.6.8/13-56
0x2_452C	MII Mgmt: control (MIIMCON)	WO	0x0000_0000	13.5.3.6.9/13-57
0x2_4530	MII Mgmt: status (MIIMSTAT)	R	0x0000_0000	13.5.3.6.10/13-57
0x2_4534	MII Mgmt: indicators (MIIMIND)	R	0x0000_0000	13.5.3.6.11/13-58
0x2_4538	Reserved	R	0x0000_0000	—
0x2_453C	Interface status (IFSTAT)	R,R/LH	0x0000_0001	13.5.3.6.12/13-58
0x2_4540	Station Address, part 1 (MACSTNADDR1)	R/W	0x0000_0000	13.5.3.6.13/13-59
0x2_4544	Station Address, part 2 (MACSTNADDR2)	R/W	0x0000_0000	13.5.3.6.14/13-60
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	13.5.3.7.1/13-61
0x2_4684	TR127 transmit and receive 65- to 127-byte frame counter	R/W	0x0000_0000	13.5.3.7.2/13-62
0x2_4688	TR255 transmit and receive 128- to 255-byte frame counter	R/W	0x0000_0000	13.5.3.7.3/13-62
0x2_468C	TR511 transmit and receive 256- to 511-byte frame counter	R/W	0x0000_0000	13.5.3.7.4/13-63
0x2_4690	TR1K transmit and receive 512- to 1023-byte frame counter	R/W	0x0000_0000	13.5.3.7.5/13-64
0x2_4694	TRMAX transmit and receive 1024- to 1518-byte frame counter	R/W	0x0000_0000	13.5.3.7.6/13-64
0x2_4698	TRMGV transmit and receive 1519- to 1522-byte good VLAN frame count	R/W	0x0000_0000	13.5.3.7.7/13-65
TSEC1 Receive Counters				
0x2_469C	RBYT receive byte counter	R/W	0x0000_0000	13.5.3.7.8/13-66
0x2_46A0	RPKT receive packet counter	R/W	0x0000_0000	13.5.3.7.9/13-66
0x2_46A4	RFCS receive FCS error counter	R/W	0x0000_0000	13.5.3.7.10/13-67
0x2_46A8	RMCA receive multicast packet counter	R/W	0x0000_0000	13.5.3.7.11/13-68

Table 13-3. Module Memory Map (continued)

Offset	Name	Access [1]	Reset	Section/Page
0x2_46AC	RBCA receive broadcast packet counter	R/W	0x0000_0000	13.5.3.7.12/13-68
0x2_46B0	RXCF receive control frame packet counter	R/W	0x0000_0000	13.5.3.7.13/13-69
0x2_46B4	RXPF receive PAUSE frame packet counter	R/W	0x0000_0000	13.5.3.7.14/13-70
0x2_46B8	RXUO receive unknown OP code counter	R/W	0x0000_0000	13.5.3.7.15/13-70
0x2_46BC	RALN receive alignment error counter	R/W	0x0000_0000	13.5.3.7.16/13-71
0x2_46C0	RFLR receive frame length error counter	R/W	0x0000_0000	13.5.3.7.17/13-72
0x2_46C4	RCDE receive code error counter	R/W	0x0000_0000	13.5.3.7.18/13-72
0x2_46C8	RCSE receive carrier sense error counter	R/W	0x0000_0000	13.5.3.7.19/13-73
0x2_46CC	RUND receive undersize packet counter	R/W	0x0000_0000	13.5.3.7.20/13-74
0x2_46D0	ROVR receive oversize packet counter	R/W	0x0000_0000	13.5.3.7.21/13-74
0x2_46D4	RFRG receive fragments counter	R/W	0x0000_0000	13.5.3.7.22/13-75
0x2_46D8	RJBR receive jabber counter	R/W	0x0000_0000	13.5.3.7.23/13-76
0x2_46DC	RDRP receive drop	R/W	0x0000_0000	13.5.3.7.24/13-76
Transmit Counters				
0x2_46E0	TBYT transmit byte counter	R/W	0x0000_0000	13.5.3.7.25/13-77
0x2_46E4	TPKT transmit packet counter	R/W	0x0000_0000	13.5.3.7.26/13-78
0x2_46E8	TMCA transmit multicast packet counter	R/W	0x0000_0000	13.5.3.7.27/13-78
0x2_46EC	TBCA transmit broadcast packet counter	R/W	0x0000_0000	13.5.3.7.28/13-79
0x2_46F0	TXPF transmit PAUSE control frame counter	R/W	0x0000_0000	13.5.3.7.29/13-80
0x2_46F4	TDFR transmit deferral packet counter	R/W	0x0000_0000	13.5.3.7.30/13-80
0x2_46F8	TEDF transmit excessive deferral packet counter	R/W	0x0000_0000	13.5.3.7.31/13-81
0x2_46FC	TSCL transmit single collision packet counter	R/W	0x0000_0000	13.5.3.7.32/13-82
0x2_4700	TMCL transmit multiple collision packet counter	R/W	0x0000_0000	13.5.3.7.33/13-82
0x2_4704	TLCL transmit late collision packet counter	R/W	0x0000_0000	13.5.3.7.34/13-83
0x2_4708	TXCL transmit excessive collision packet counter	R/W	0x0000_0000	13.5.3.7.35/13-84
0x2_470C	TNCL transmit total collision counter	R/W	0x0000_0000	13.5.3.7.36/13-84
0x2_4710	Reserved	R	0x0000_0000	—
0x2_4714	TDRP transmit drop frame counter	R/W	0x0000_0000	13.5.3.7.37/13-85
0x2_4718	TJBR transmit jabber frame counter	R/W	0x0000_0000	13.5.3.7.38/13-86
0x2_471C	TFCS transmit FCS error counter	R/W	0x0000_0000	13.5.3.7.39/13-86
0x2_4720	TXCF transmit control frame counter	R/W	0x0000_0000	13.5.3.7.40/13-87
0x2_4724	TOVR transmit oversize frame counter	R/W	0x0000_0000	13.5.3.7.41/13-88
0x2_4728	TUND transmit undersize frame counter	R/W	0x0000_0000	13.5.3.7.42/13-88

Table 13-3. Module Memory Map (continued)

Offset	Name	Access [1]	Reset	Section/Page
0x2_472C	TFRG transmit fragments frame counter	R/W	0x0000_0000	13.5.3.7.43/13-89
General Registers				
0x2_4730	CAR1 carry register one register [2]	R	0x0000_0000	13.5.3.7.44/13-90
0x2_4734	CAR2 carry register two register [2]	R	0x0000_0000	13.5.3.7.45/13-91
0x2_4738	CAM1 carry register one mask register	R/W	0xFE01_FFFF	13.5.3.7.46/13-92
0x2_473C	CAM2 carry register two mask register	R/W	0x000F_FFFF	13.5.3.7.47/13-93
0x2_4740–0x2_47FC	Reserved	R	0x0000_0000	—
Hash Function Registers				
0x2_4800	Individual address register 0 (IADDR0)	R/W	0x0000_0000	13.5.3.8.1/13-95
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	13.5.3.8.2/13-95
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	13.5.3.8.3/13-96
0x2_4BFC	Attribute EL & EI register (ATTRELI)	R	0x0000_0000	13.5.3.8.4/13-97
TSEC1 Future Expansion Space				

Table 13-3. 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]			13.5.3/13-21

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 0x 2_4000 to 0x2_4FFF except the offsets are from 0x 2_5000 to 0x2_5FFF.

13.5.3 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.

13.5.3.1 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.

13.5.3.1.1 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 13-8 describes the definition for the IEVENT register.

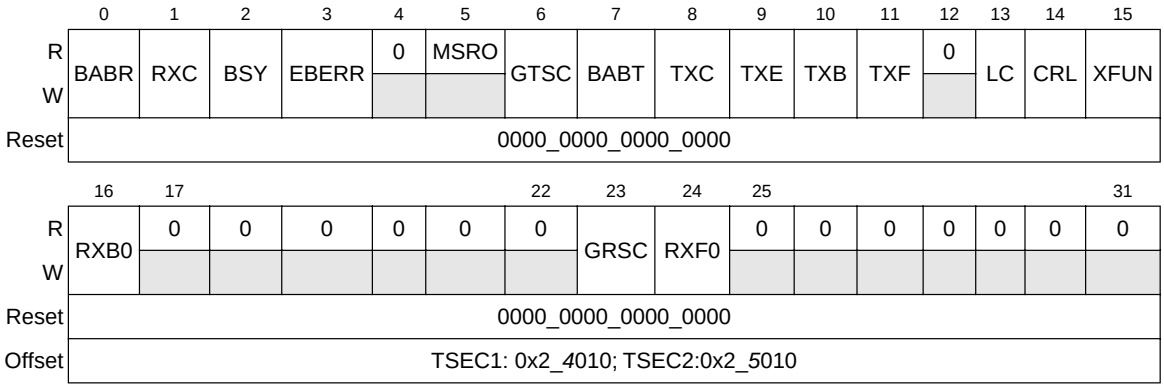


Figure 13-6. IEVENT Register Definition

Table 13-4 describes the fields of the IEVENT register.

Table 13-4. 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 13-4. 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 13-4. 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

13.5.3.1.2 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 13-7 describes the definition for the IMASK register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BREN	RXCEN	BSYEN	EBERREN	0	MSROEN	GTSCEN	BTEN	TXCEN	TXEEN	TXBEN	TXFEN	0	LCEN	CRLEN	XFUNEN
W																
Reset	0000_0000_0000_0000															

	16	17					22	23	24	25						31
R	RXBEN0	0	0	0	0	0	0	GRSCEN	RXFEN0	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4014; TSEC2:0x2_5014															

Figure 13-7. IMASK Register Definition

Table 13-5 describes the fields of the IMASK register.

Table 13-5. IMASK Field Descriptions

Bits	Name	Description
0	BREN	Babbling receiver interrupt enable
1	RXCEN	Receive control interrupt enable

Table 13-5. 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

13.5.3.1.3 Error Disabled Register (EDIS)

Figure 13-8 describes the definition for the EDIS register. The error disabled register allows the user to disable an error interruption.

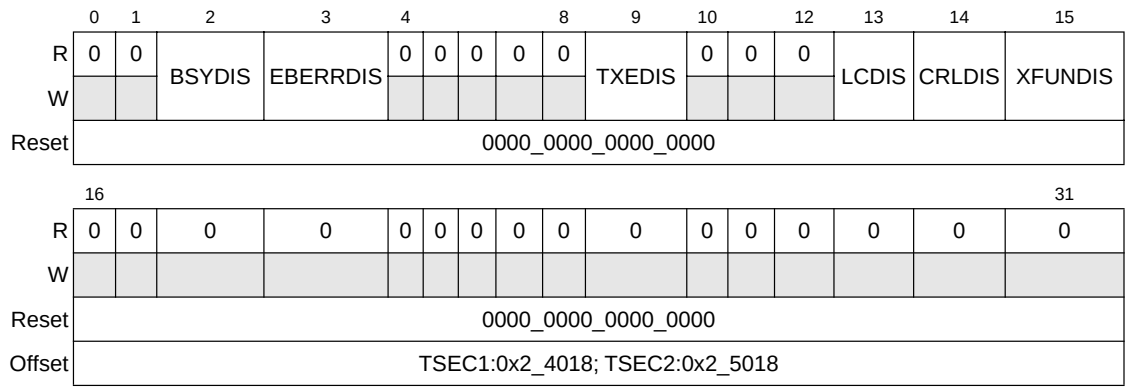


Figure 13-8. EDIS Register Definition

Table 13-6 describes the fields of the EDIS register.

Table 13-6. 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

13.5.3.1.4 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 13-9 describes the definition for the ECNTRL register.

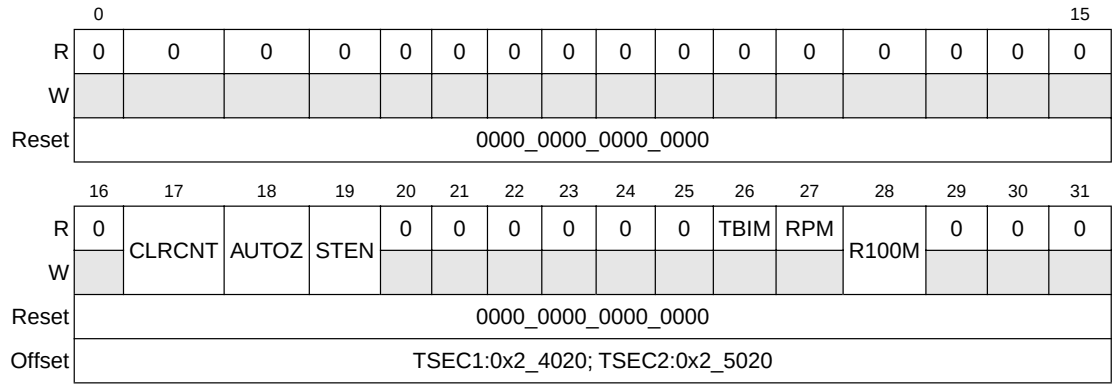


Figure 13-9. ECNTRL Register Definition

Table 13-7 describes the fields of the ECNTRL register.

Table 13-7. 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

13.5.3.1.5 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 13-10 describes the definition for the MINFLR register.

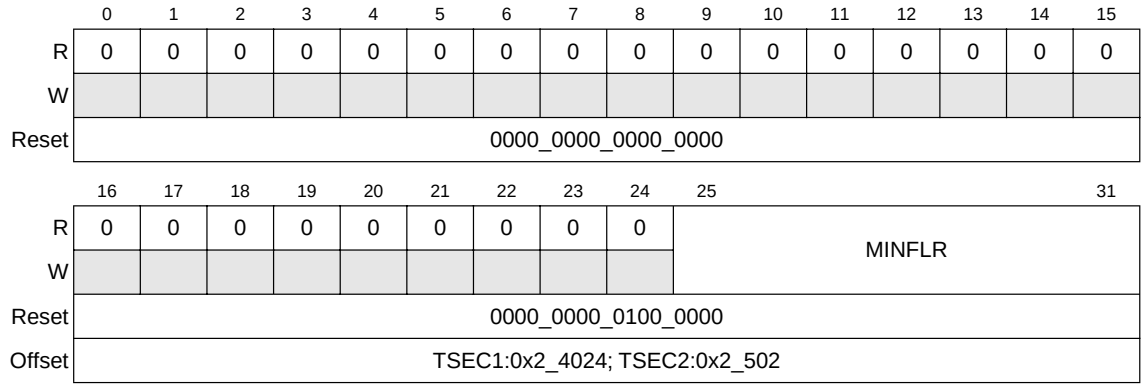


Figure 13-10. MINFLR Register Definition

Table 13-8 describes the fields of the MINFLR register.

Table 13-8. 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.

13.5.3.1.6 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 13.6.2.7, “Flow Control,” on page 13-126 for additional details. Figure 13-11 describes the definition for the PTV register.

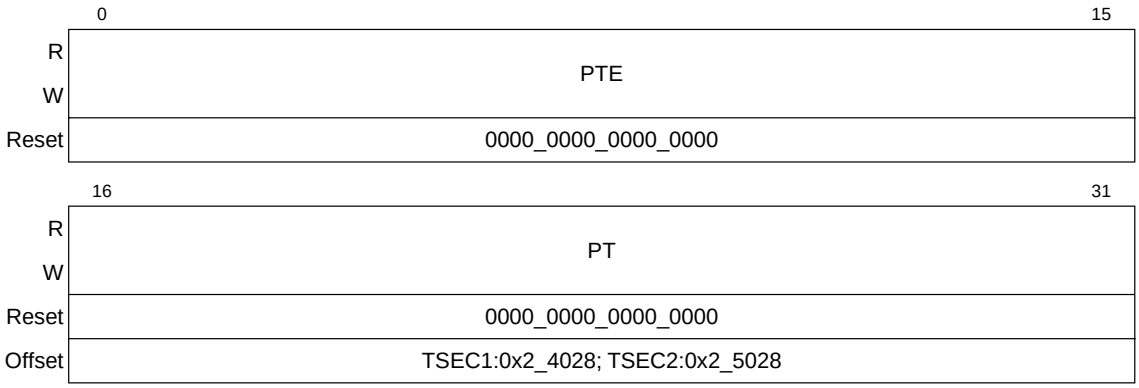


Figure 13-11. PTV Register Definition

Table 13-9 describes the fields of the PTV register.

Table 13-9. PTV Field Descriptions

Bits	Name	Description
0–15	PTE	Extended pause time value
16–31	PT	Pause time value

13.5.3.1.7 DMA Control Register (DMACTRL)

DMACTRL is writable by the user to configure the DMA block. Figure 13-12 describes the definition for the DMACTRL register.

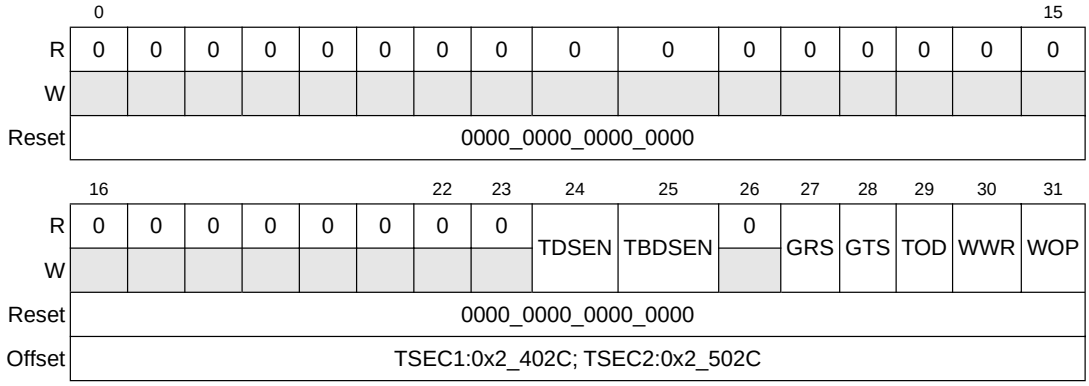


Figure 13-12. DMACTRL Register Definition

Table 13-10 describes the fields of the DMACTRL register.

Table 13-10. 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].

13.5.3.1.8 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 13.5.4, “Ten-Bit Interface (TBI).” Figure 13-13 describes the definition for the TBIPA register.

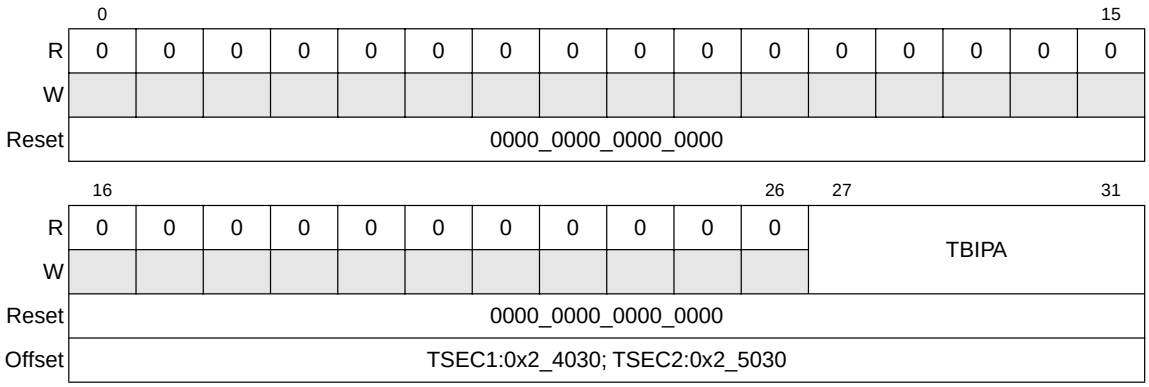


Figure 13-13. TBIPA Register Definition

Table 13-11 describes the fields of the TBIPA register.

Table 13-11. 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 13.5.3.6.8, “MII Management Address Register (MIIMADD).”

13.5.3.2 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 indicates 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.

13.5.3.2.1 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 13-14 describes the definition for the FIFO_TX_THR register.

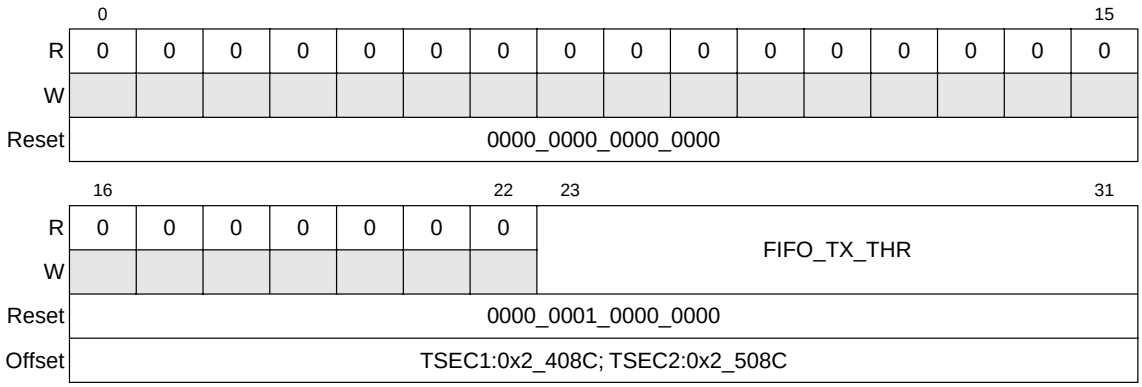


Figure 13-14. FIFO_TX_THR Register Definition

Table 13-12 describes the fields of the FIFO_TX_THR register.

Table 13-12. 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.

13.5.3.2.2 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 13-15 describes the definition for the FIFO_TX_STARVE register.

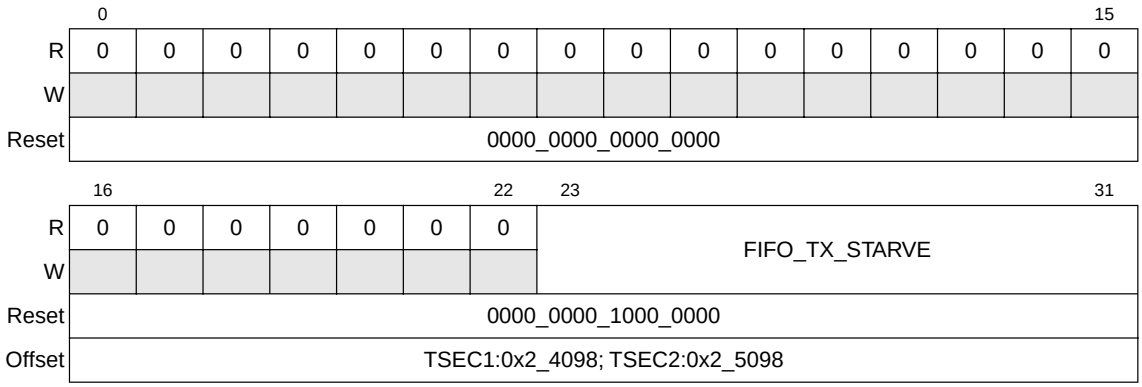


Figure 13-15. FIFO_TX_STARVE Register Definition

Table 13-13 describes the fields of the FIFO_TX_STARVE register.

Table 13-13. 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.

13.5.3.2.3 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 13-16 describes the definition for the FIFO_TX starve shutoff register.

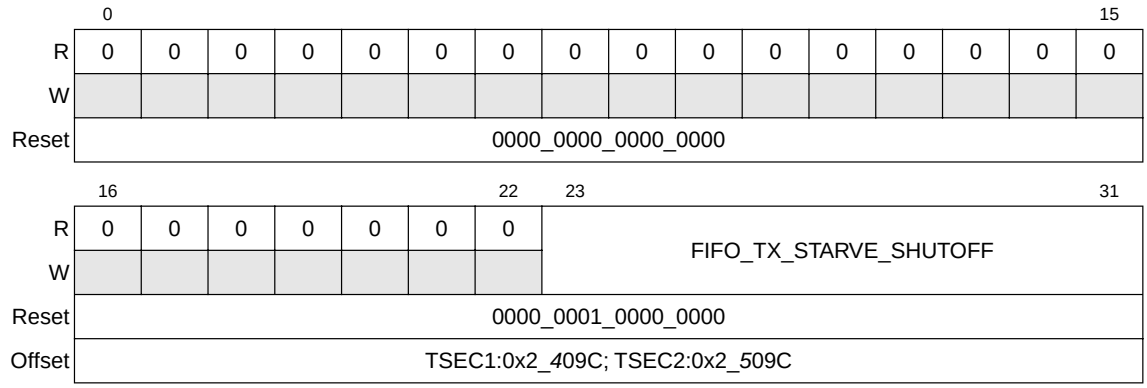


Figure 13-16. FIFO_TX_STARVE_SHUTOFF Register Definition

Table 13-14 describes the fields of the FIFO_TX starve shutoff register.

Table 13-14. 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.

13.5.3.3 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.

13.5.3.3.1 Transmit Control Register (TCTRL)

This register is writable by the user to configure the transmit block. Figure 13-17 describes the definition for the TCTRL register.

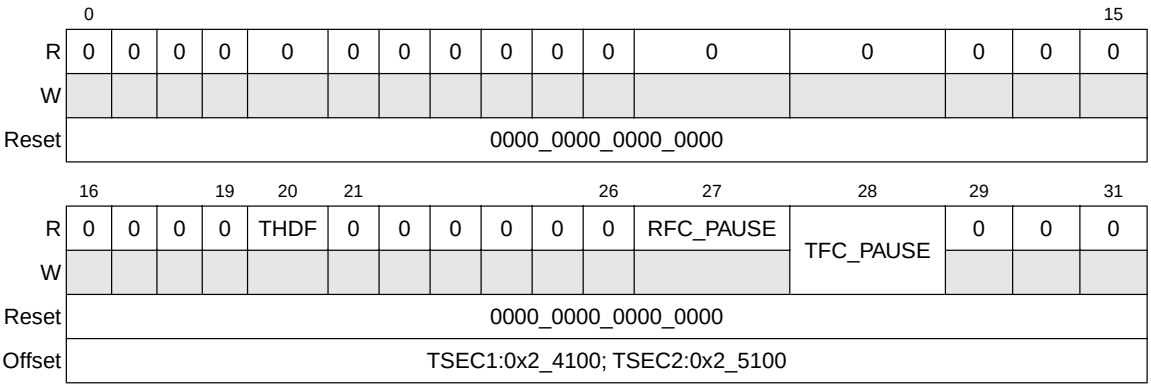


Figure 13-17. TCTRL Register Definition

Table 13-15 describes the fields of the TCTRL register.

Table 13-15. 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

13.5.3.3.2 Transmit Status Register (TSTAT)

This register is read/write and is written by TSEC to convey DMA status information. Figure 13-18 describes the definition for the TSTAT register.

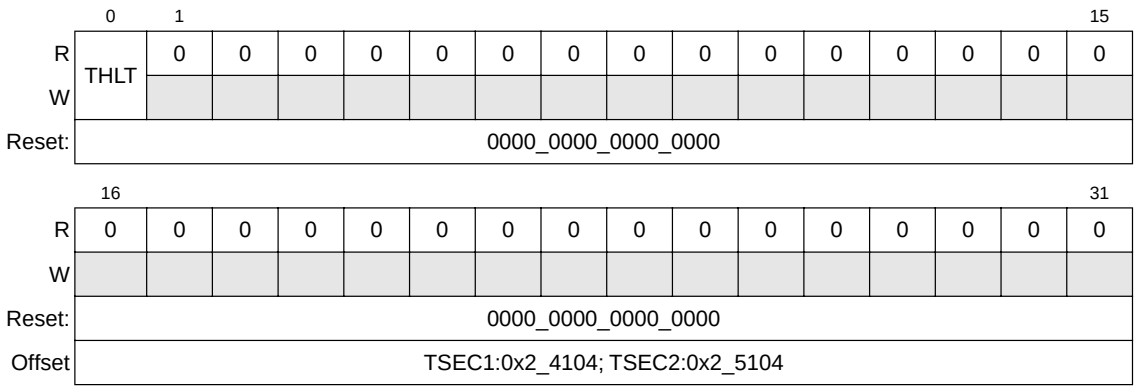


Figure 13-18. TSTAT Register Definition

Table 13-16 describes the fields of the TSTAT register.

Table 13-16. 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

13.5.3.3.3 TxBD Data Length Register (TBDLEN)

TBDLEN is a DMA register that contains the number of bytes remaining in the current transmit buffer. Figure 13-19 describes the definition for the TBDLEN 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 W	TBDLEN	
Reset	0000_0000_0000_0000	
Offset	TSEC1:0x2_410C; TSEC2:0x2_510C	

Figure 13-19. TBDLEN Register Definition

Table 13-17 describes the fields of the TBDLEN register.

Table 13-17. 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.

13.5.3.3.4 Current Transmit Buffer Descriptor Pointer (CTBPTR)

CTBPTR contains the address of the transmit buffer descriptor either currently being processed, or processed most recently. Figure 13-20 describes the definition for the CTBPTR register.

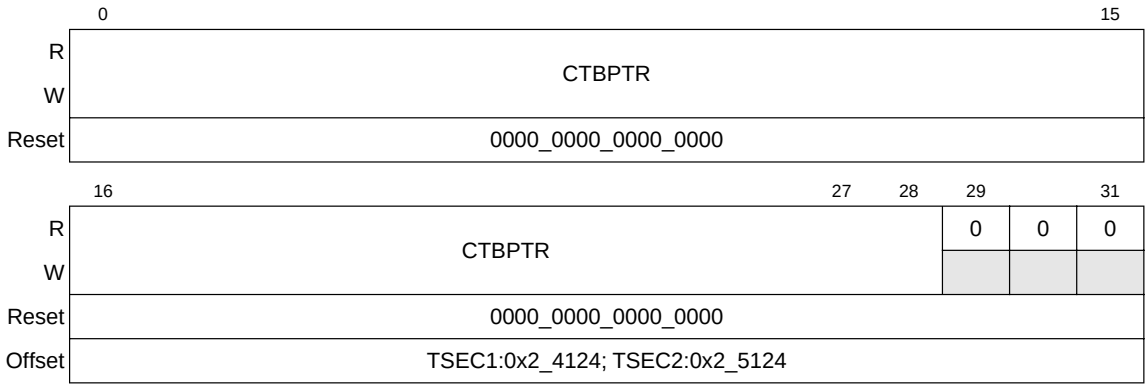


Figure 13-20. CTBPTR Register Definition

Table 13-18 describes the fields of the CTBPTR register.

Table 13-18. 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

13.5.3.3.5 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 13-21 describes the definition for the TBASE register.

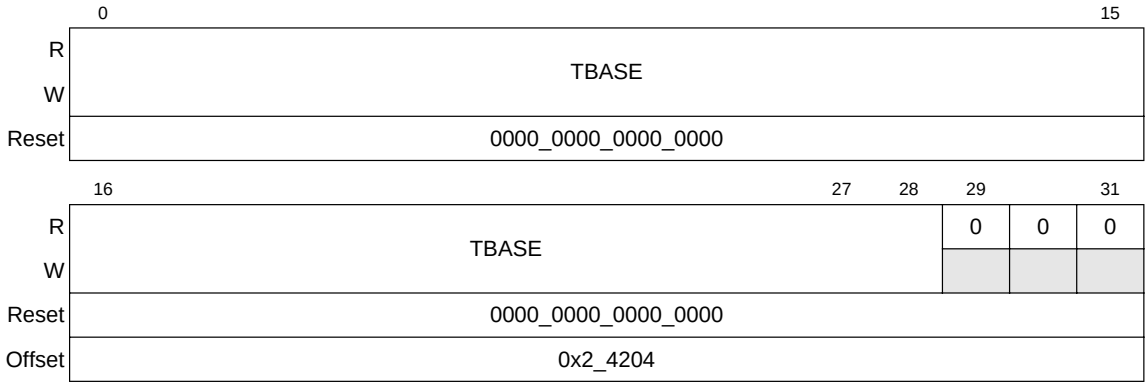


Figure 13-21. TBASE Register Definition

Table 13-19 describes the fields of the TBASE register.

Table 13-19. 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

13.5.3.3.6 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 13-22 describes the definition for the OSTBD register.

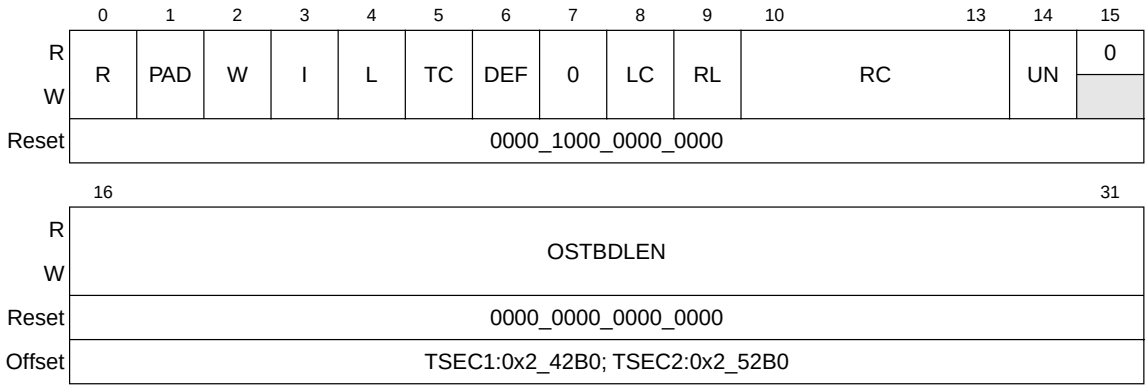


Figure 13-22. OSTBD Register Definition

Table 13-20 describes the fields of the OSTBD register.

Table 13-20. 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 where 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 13-20. 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.

13.5.3.3.7 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 13-23 describes the definition for the OSTBDP register.

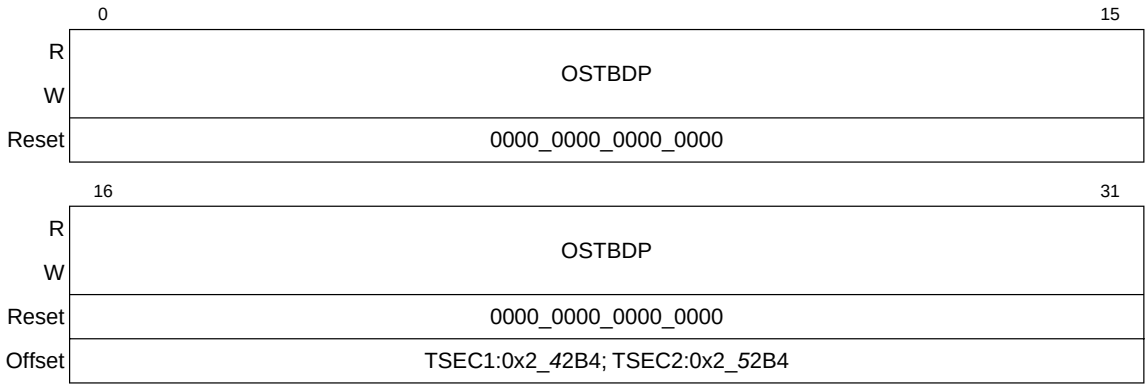


Figure 13-23. OSTBDP Register Definition

Table 13-21 describes the fields of the OSTBDP register.

Table 13-21. 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.

13.5.3.4 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.

13.5.3.4.1 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 13-24 describes the definition for the RCTRL register.

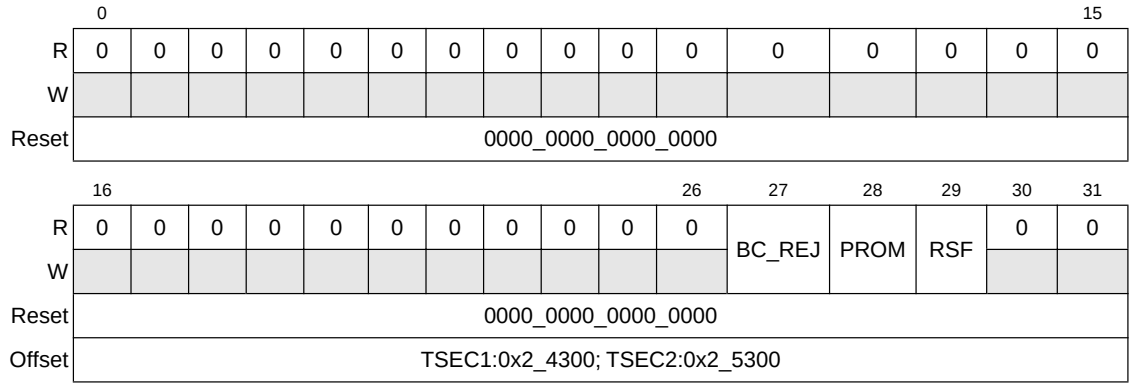


Figure 13-24. RCTRL Register Definition

Table 13-22 describes the fields of the RCTRL register.

Table 13-22. 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

13.5.3.4.2 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 13-25 describes the definition for the RSTAT register.

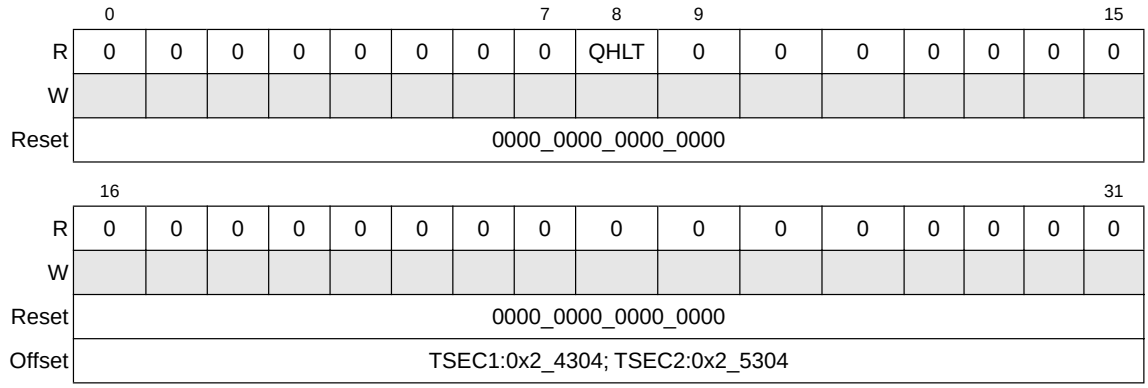


Figure 13-25. RSTAT Register Definition

Table 13-23 describes the fields of the RSTAT register.

Table 13-23. 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

13.5.3.4.3 RxBD Data Length Register (RBDLEN)

RBDLEN is a DMA register that contains the number of bytes remaining in the current receive buffer. Figure 13-26 describes the definition for the RBDLEN register.

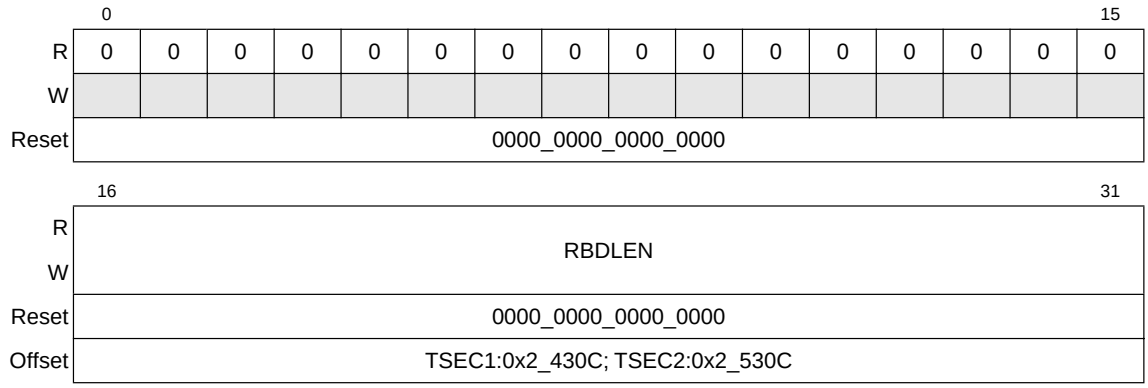


Figure 13-26. RBDLEN Register Definition

Table 13-24 describes the fields of the RBDLEN register.

Table 13-24. 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.

13.5.3.4.4 Current Receive Buffer Descriptor Pointer (CRBPTR)

CRBPTR contains the address of the receive buffer descriptor either currently being processed, or processed most recently. Figure 13-27 describes the definition for the CRBPTR register. This register is writable, but should never be written by user.

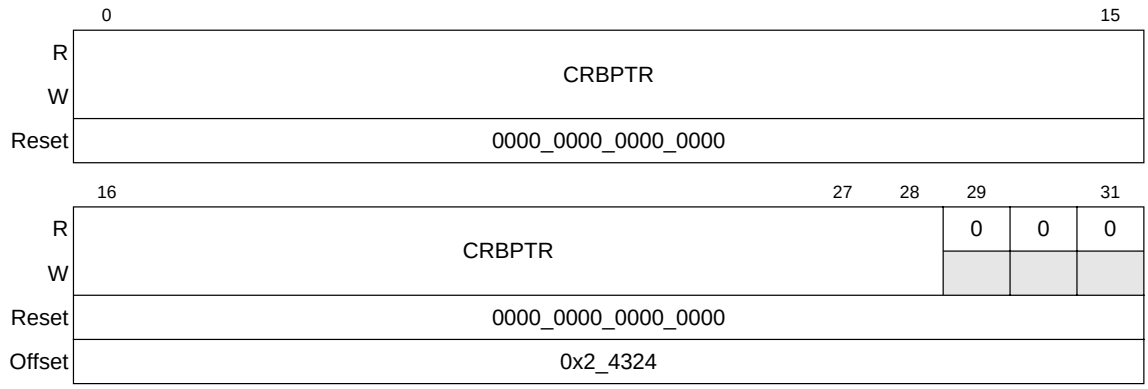


Figure 13-27. CRBPTR Register Definition

Table 13-25 describes the fields of the CRBPTR register.

Table 13-25. 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

13.5.3.4.5 Maximum Receive Buffer Length Register (MRBLR)

The MRBL register is written by the user. It informs the TSEC how much space is in the receive buffer pointed to by the RxBD. Figure 13-28 describes the definition for the MRBLR.

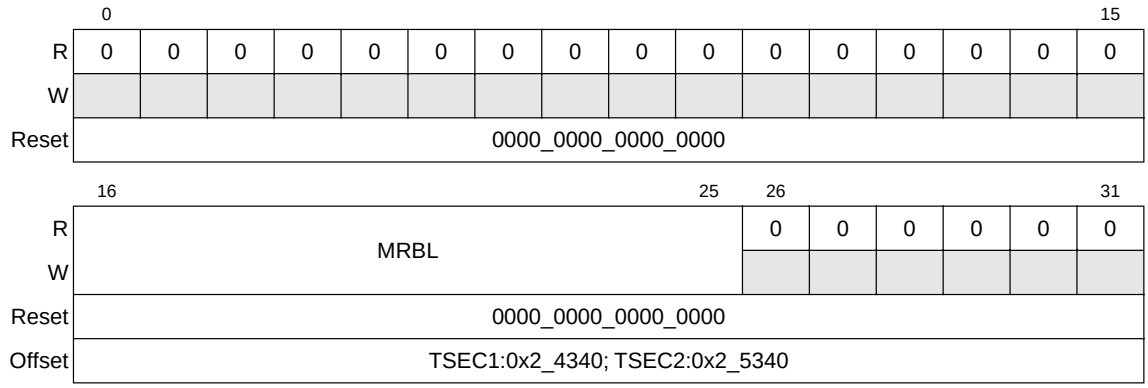


Figure 13-28. MRBL Register Definition

Table 13-26. 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.

13.5.3.4.6 Receive Buffer Descriptor Pointer (RBPTR)

RBPTR contains the receive buffer descriptor address. Figure 13-29 describes the definition for the RBPTR register. This register takes on the value of RBASE when the RBASE register is written by software.

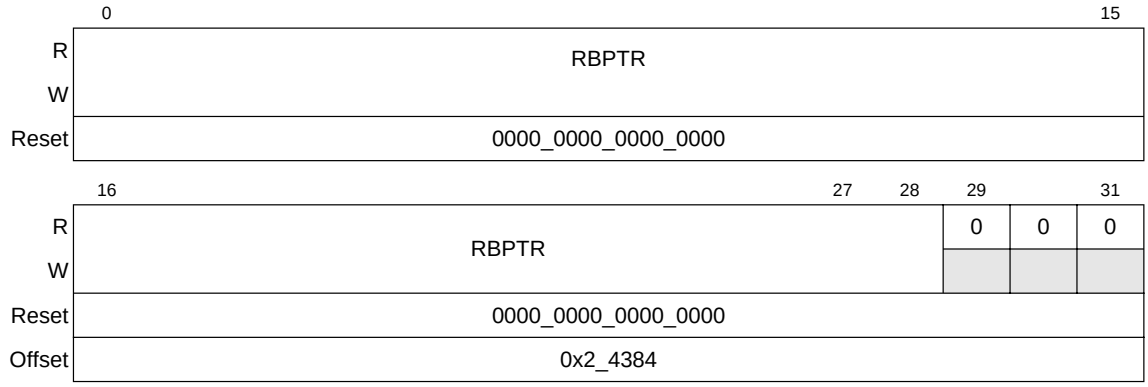


Figure 13-29. Receive Buffer Descriptor Pointer Definition

Table 13-27 describes the fields of the RBPTR register.

Table 13-27. 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

13.5.3.4.7 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 13-30 describes the definition for the RBASE register.

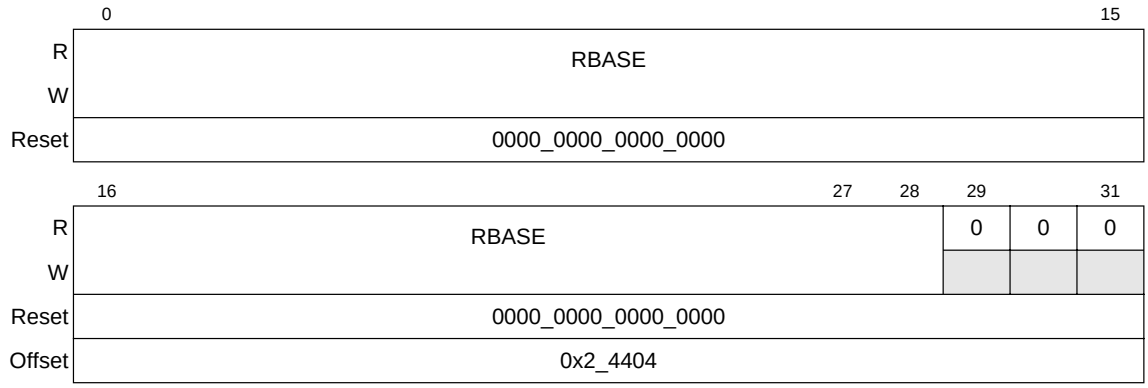


Figure 13-30. RBASE Register Definition

Table 13-28 describes the fields of the RBASE register.

Table 13-28. 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

13.5.3.5 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.

13.5.3.5.1 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.

13.5.3.5.2 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.

13.5.3.5.3 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:

$0 < \text{or} = r < \text{or} = 2^k$ 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.

13.5.3.5.4 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.

13.5.3.5.5 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 indicatorregisters) 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.

13.5.3.6 MAC Registers

Following are the MAC registers.

13.5.3.6.1 MAC Configuration 1 Register (MACCFG1)

The MACCFG1 register is written by the user. Figure 13-31 describes the definition for the MACCFG1 register.

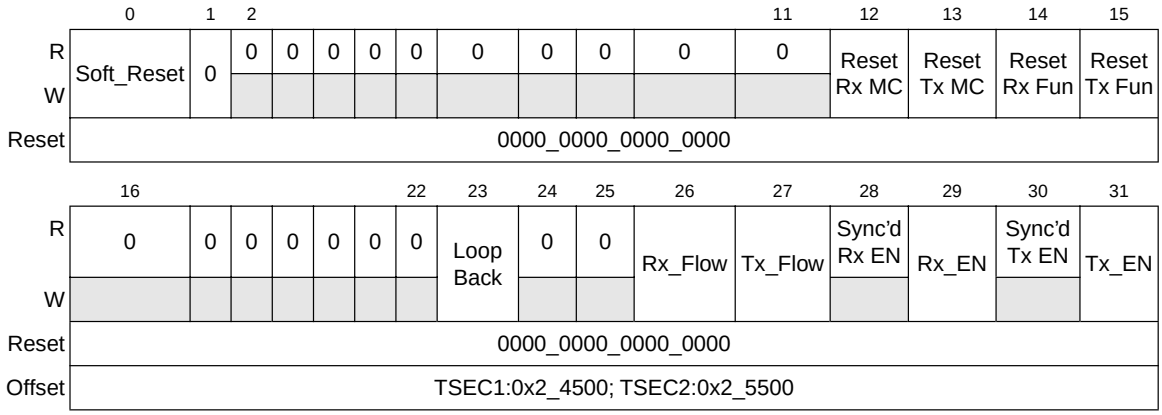


Figure 13-31. MACCFG1 Register Definition

Table 13-29 describes the fields of the MACCFG1 register.

Table 13-29. MACCFG1 Field Descriptions

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. See Section 13.6.2.2, “Soft Reset and Reconfiguring Procedure,” on page 13-119 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 13-29. 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.

13.5.3.6.2 MAC Configuration 2 Register (MACCFG2)

The MACCFG2 register is written by the user. Figure 13-32 describes the definition for the MACCFG2 register.

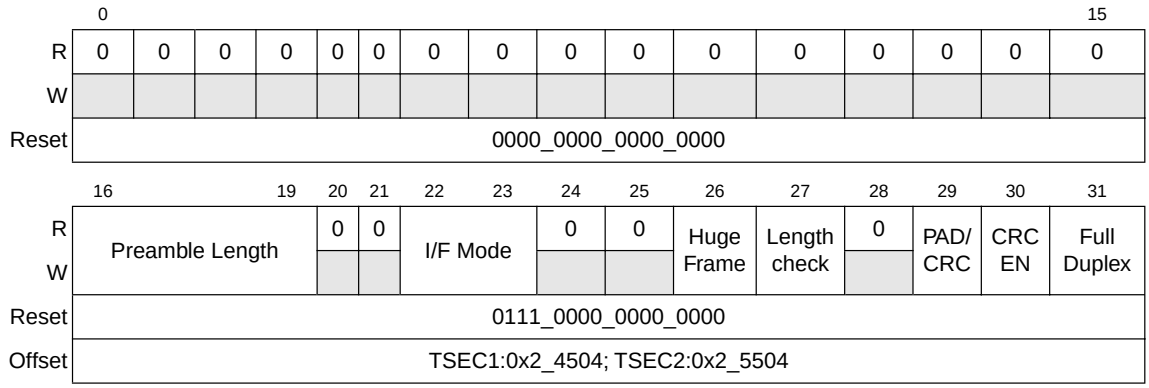


Figure 13-32. MACCFG2 Register Definition

Table 13-30 describes the fields of the MACCFG2 register.

Table 13-30. 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.

13.5.3.6.3 Inter-Packet Gap/Inter-Frame Gap Register (IPGIFG)

The IPGIFG register is written by the user. Figure 13-33 describes the definition for the IPGIFG register.

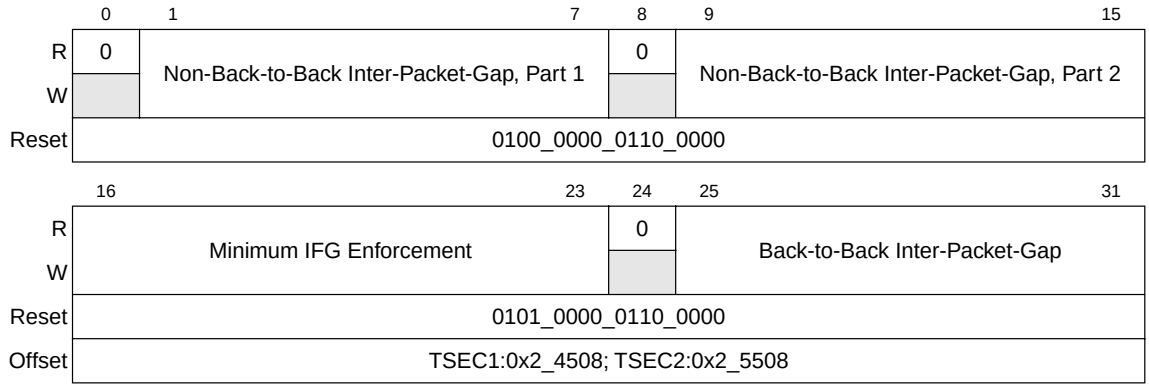


Figure 13-33. IPGIFG Register Definition

Table 13-31 describes the fields of the IPGIFG register.

Table 13-31. 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.

13.5.3.6.4 Half-Duplex Register (HAFDUP)

The HAFDUP register is written by the user. Figure 13-34 describes the definition for the HAFDUP register.

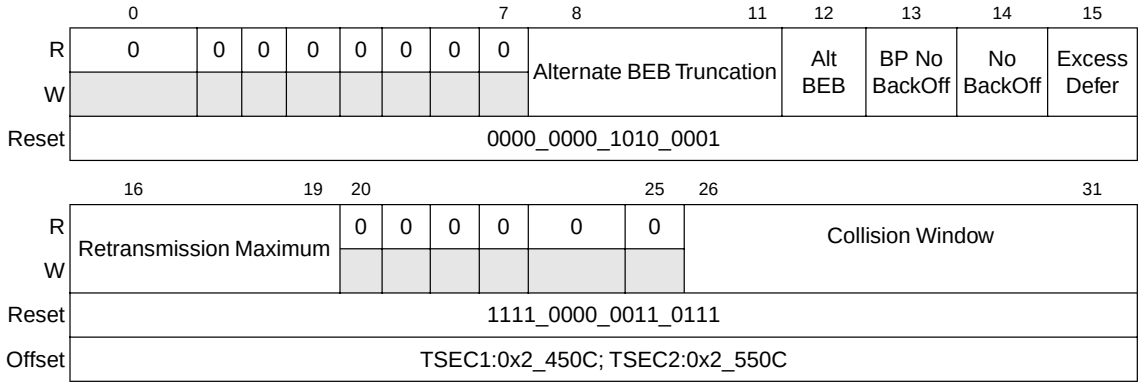


Figure 13-34. Half-Duplex Register Definition

Table 13-32 describes the fields of the HAFDUP register.

Table 13-32. 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 210 as the maximum backoff time.
13	BP No BackOff	Back pressure no backoff. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back off rule. 1 The Tx MAC immediately re-transmits, following a collision, during back pressure operation.
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.

13.5.3.6.5 Maximum Frame Length Register (MAXFRM)

The MAXFRM register is written by the user. Figure 13-35 describes the definition for the MAXFRM register.

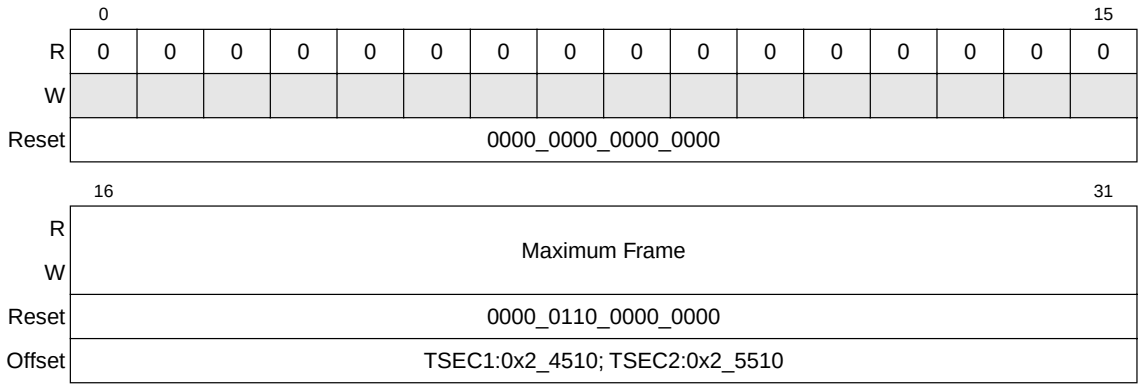


Figure 13-35. Maximum Frame Length Register Definition

Table 13-33 describes the fields of the MAXFRM register.

Table 13-33. 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].)

13.5.3.6.6 MII Management Configuration Register (MIIMCFG)

The MIIMCFG register is written by the user. Figure 13-36 describes the definition for the MIIMCFG register.

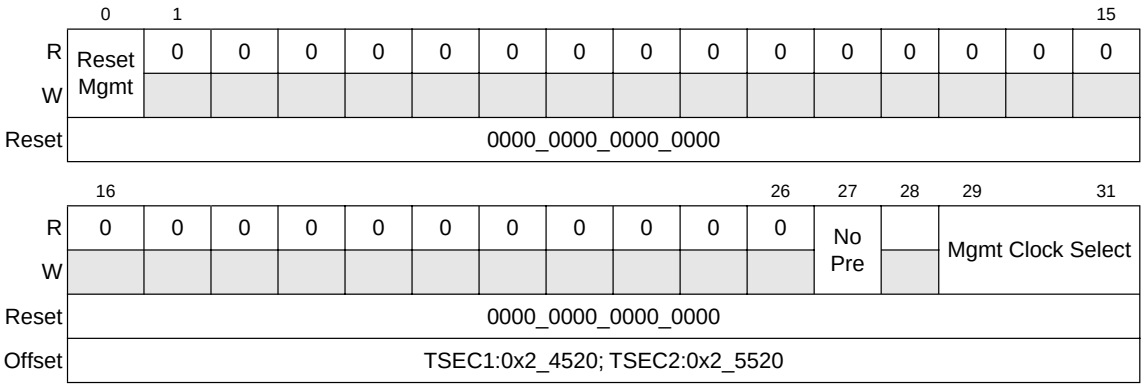


Figure 13-36. MII Management Configuration Register Definition

Table 13-34 describes the fields of the MIIMCFG register.

Table 13-34. 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

13.5.3.6.7 MII Management Command (MIIMCOM)

The MIIMCOM register is written by the user. Figure 13-37 describes the definition for the MIIMCOM register.

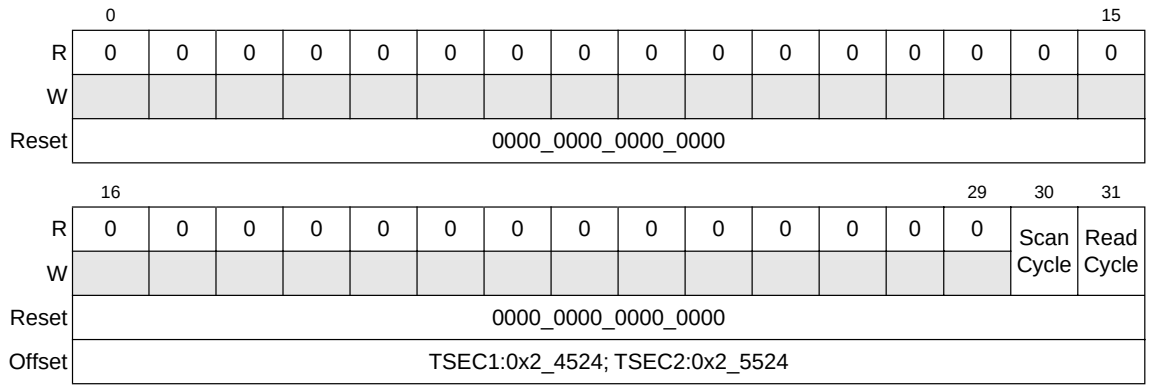


Figure 13-37. MIIMCOM Register Definition

Table 13-35 describes the fields of the MIIMCOM register.

Table 13-35. 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].

13.5.3.6.8 MII Management Address Register (MIIMADD)

The MIIMADD register is written by the user. Figure 13-38 describes the definition for the MIIMADD register.

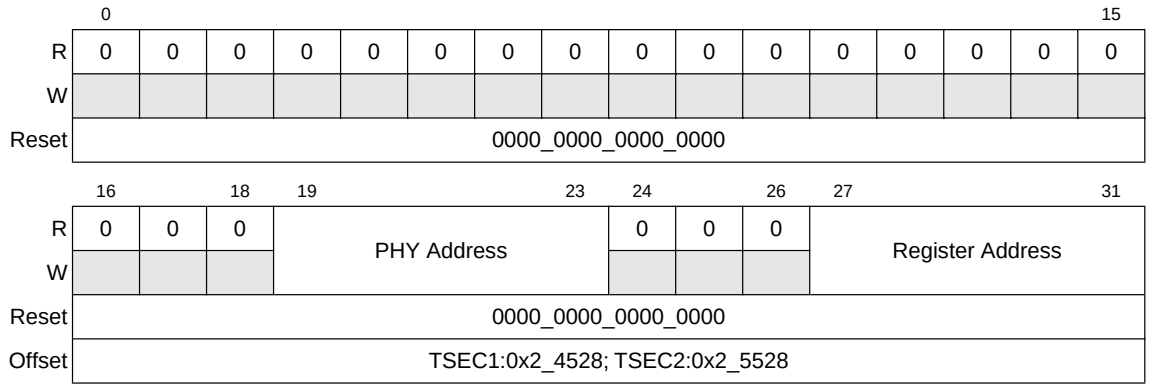


Figure 13-38. MIIMADD Register Definition

Table 13-36 describes the fields of the MIIMADD register.

Table 13-36. 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.

13.5.3.6.9 MII Management Control Register (MIIMCON)

The MIIMCON register is written by the user. Figure 13-39 describes the definition for the MIIMCON register.

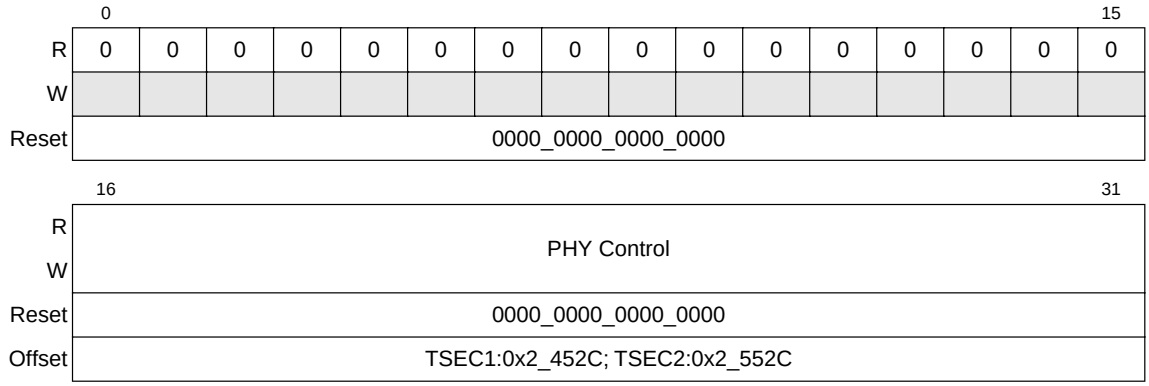


Figure 13-39. MII Mgmt Control Register Definition

Table 13-37 describes the fields of the MIIMCON register.

Table 13-37. 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.

13.5.3.6.10 MII Management Status Register (MIIMSTAT)

The MIIMSTAT register is read-only by the user. Figure 13-40 describes the definition for the MIIMSTAT register.

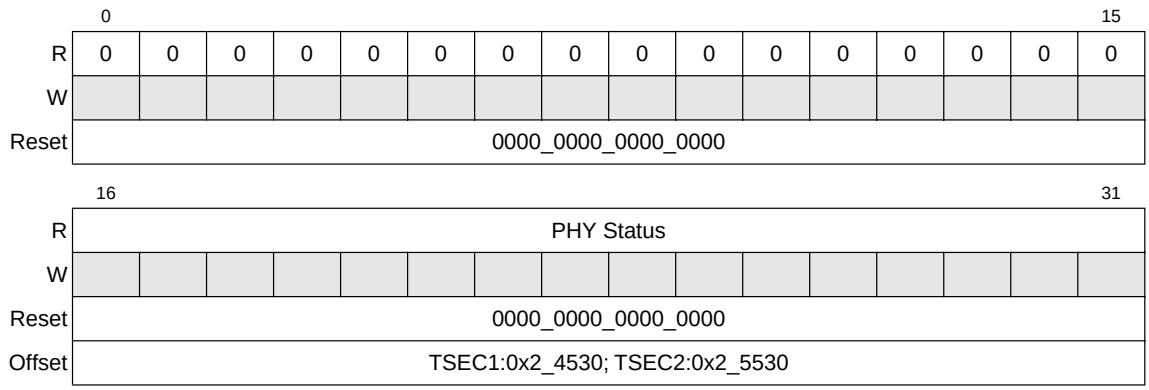


Figure 13-40. MIIMSTAT Register Definition

Table 13-38 describes the fields of the MIIMSTAT register.

Table 13-38. 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.

13.5.3.6.11 MII Management Indicator Register (MIIMIND)

The MIIMIND register is read-only by the user. Figure 13-41 describes the definition for the MIIMIND 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												28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	Not Valid	Scan	Busy
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4534; TSEC2:0x2_5534															

Figure 13-41. MII Mgmt Indicator Register Definition

Table 13-39. 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.

13.5.3.6.12 Interface Status Register (IFSTAT)

The IFSTAT register is written by the user. Figure 13-42 describes the definition for the IFSTAT register.

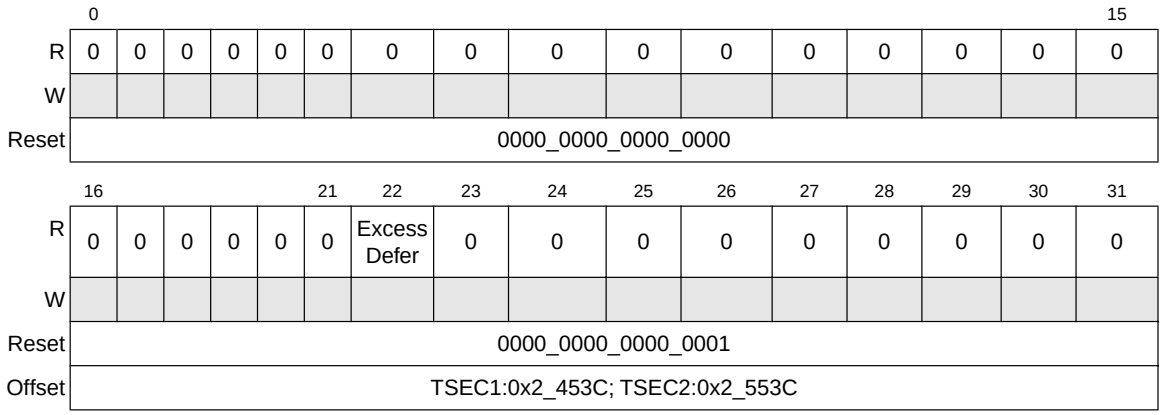


Figure 13-42. Interface Status Register Definition

Table 13-40 describes the fields of the FSTAT register.

Table 13-40. 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

13.5.3.6.13 Station Address Part 1 Register (MACSTNADDR1)

The MACSTNADDR1 register is written by the user. Figure 13-43 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.

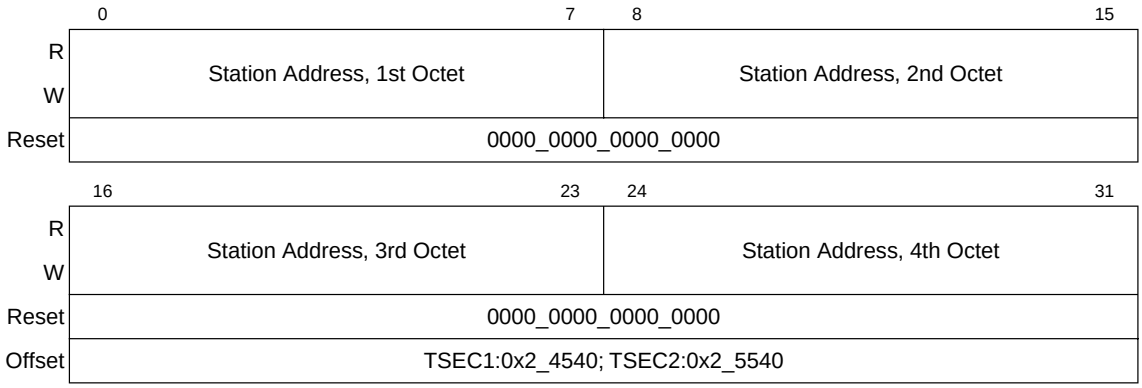


Figure 13-43. Station Address Part 1 Register Definition

Table 13-41 describes the fields of the MACSTNADR1 register.

Table 13-41. 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.

13.5.3.6.14 Station Address Part 2 Register (MACSTNADDR2)

The MACSTNADDR2 register is written by the user. Figure 13-44 describes the definition for the MACSTNADDR2 register.

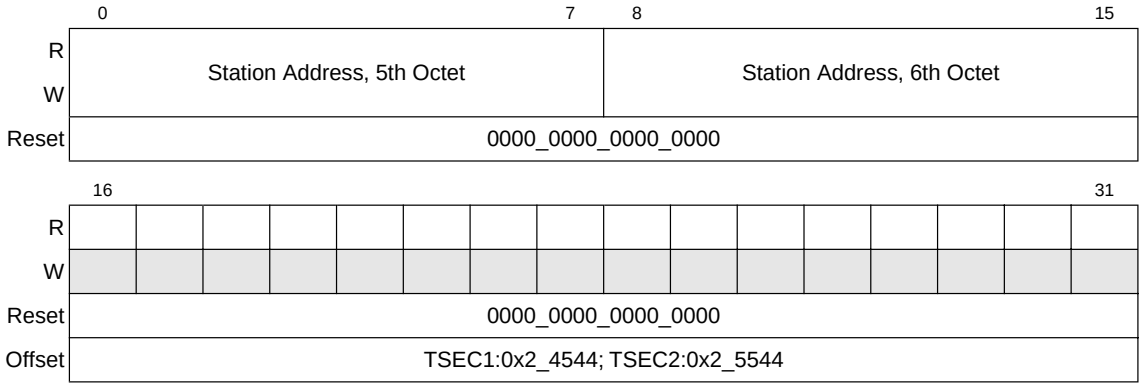


Figure 13-44. Station Address Part 2Register Definition

Table 13-42 describes the fields of the MACSTNADDR2 register.

Table 13-42. 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 8–15and defaults to a value of 0x00.
16–31	—	Reserved

13.5.3.7 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 13-45 describes the definition for the TR64 register.

13.5.3.7.1 Transmit and Receive 64-Byte Frame Counter (TR64)

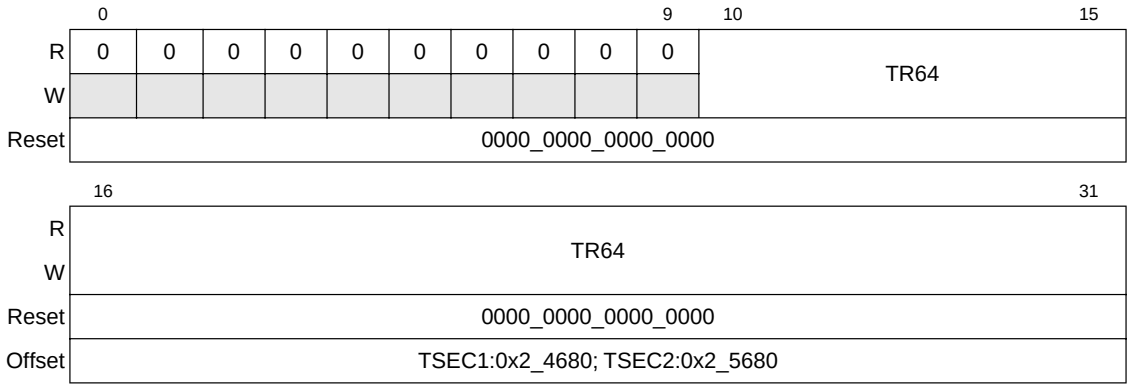


Figure 13-45. Transmit and Receive 64-Byte Frame Register Definition

Table 13-43 describes the fields of the TR64 register.

Table 13-43. 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).

13.5.3.7.2 Transmit and Receive 65- to 127-Byte Frame Counter (TR127)

Figure 13-46 describes the definition for the TR127 register.

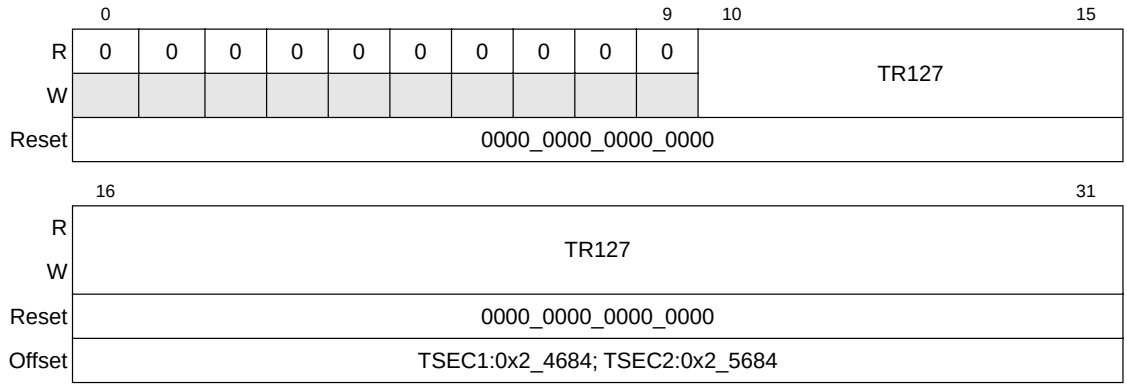


Figure 13-46. Transmit and Receive 65- to 127-Byte Frame Register Definition

Table 13-44 describes the fields of the TR127 register.

Table 13-44. 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).

13.5.3.7.3 Transmit and Receive 128- to 255-Byte Frame Counter (TR255)

Figure 13-47 describes the definition for the TR255 register.

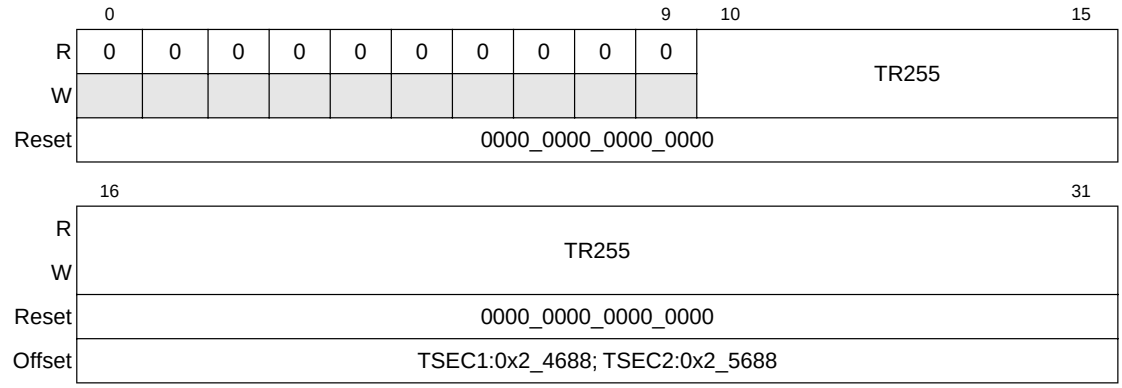


Figure 13-47. Transmit and Received 128- to 255-Byte Frame Register Definition

Table 13-45 describes the fields of the TR255 register.

Table 13-45. 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).

13.5.3.7.4 Transmit and Receive 256- to 511-Byte Frame Counter (TR511)

Figure 13-48 describes the definition for the TR511 register.

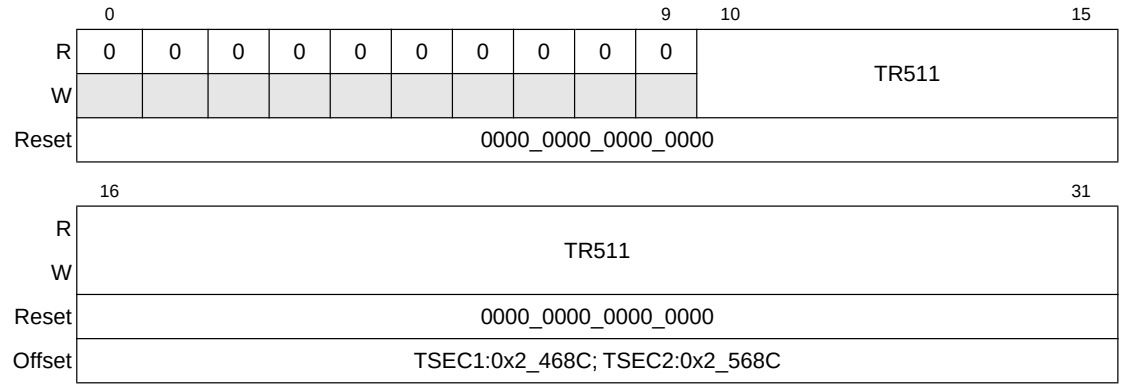


Figure 13-48. Transmit and Received 256- to 511-Byte Frame Register Definition

Table 13-46 describes the fields of the TR511 register.

Table 13-46. 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).

13.5.3.7.5 Transmit and Receive 512- to 1023-Byte Frame Counter (TR1K)

Figure 13-49 describes the definition for the TR1K register.

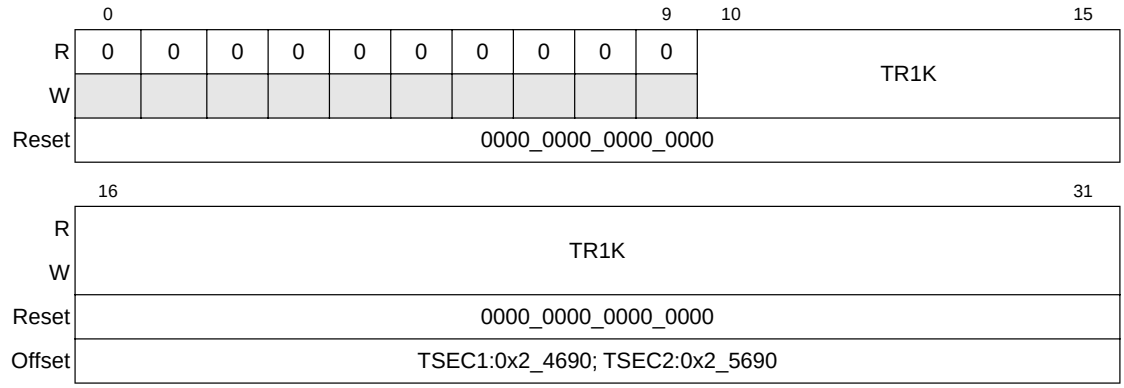


Figure 13-49. Transmit and Received 512- to 1023-Byte Frame Register Definition

Table 13-47 describes the fields of the TR1K register.

Table 13-47. 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).

13.5.3.7.6 Transmit and Receive 1024- to 1518-Byte Frame Counter (TRMAX)

Figure 13-50 describes the definition for the TRMAX register.

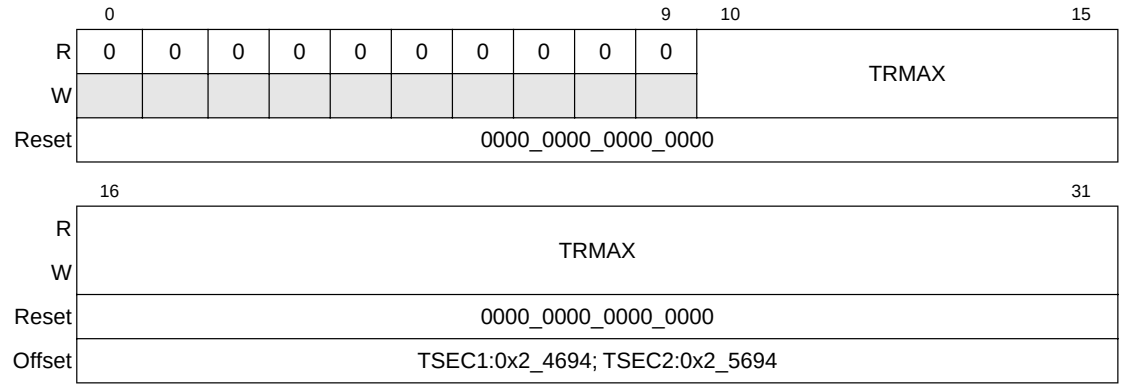


Figure 13-50. Transmit and Received 1024- to 1518-Byte Frame Register Definition

Table 13-48 describes the fields of the TRMAX register.

Table 13-48. 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).

13.5.3.7.7 Transmit and Receive 1519- to 1522-Byte VLAN Frame Counter (TRMGV)

Figure 13-51 describes the definition for the TRMGV register.

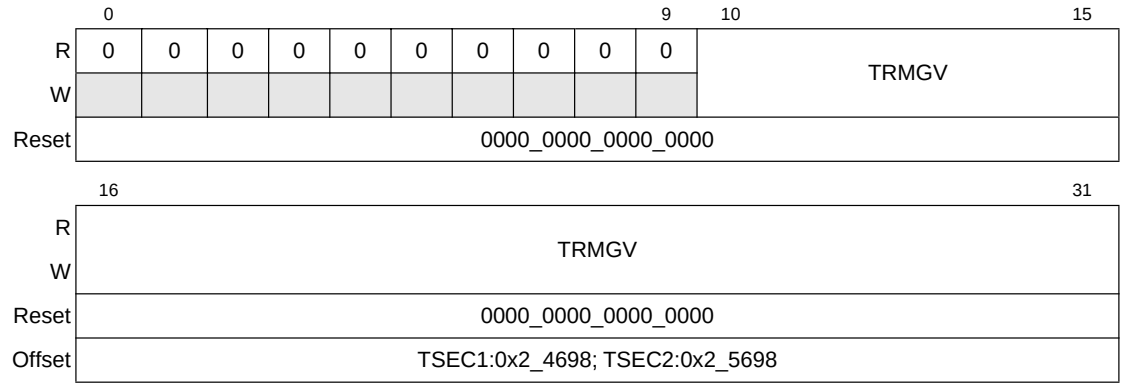


Figure 13-51. Transmit and Received 1519- to 1522-Byte VLAN Frame Register Definition

Table 13-49 describes the fields of the TRMGV register.

Table 13-49. 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).

13.5.3.7.8 Receive Byte Counter (RBYT)

Figure 13-52 describes the definition for the RBYT register.

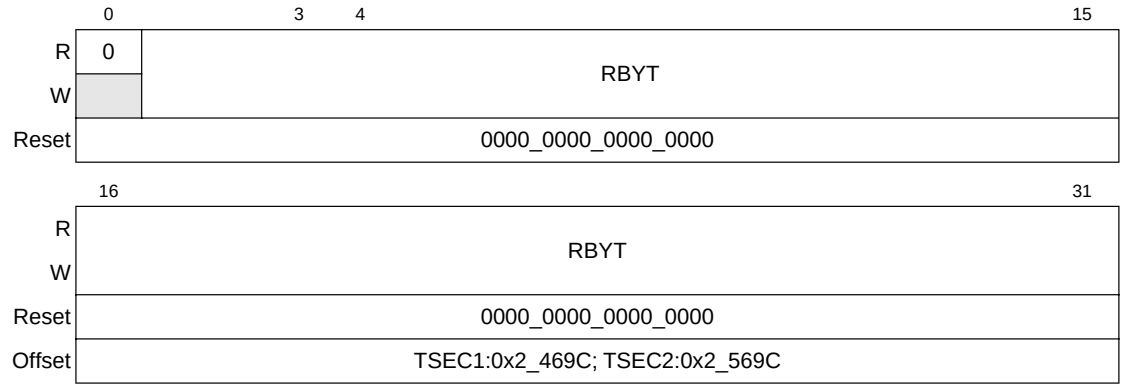


Figure 13-52. Receive Byte Counter Register Definition

Table 13-50 describes the fields of the RBYT register.

Table 13-50. 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.

13.5.3.7.9 Receive Packet Counter (RPKT)

Figure 13-53 describes the definition for the RPKT register.

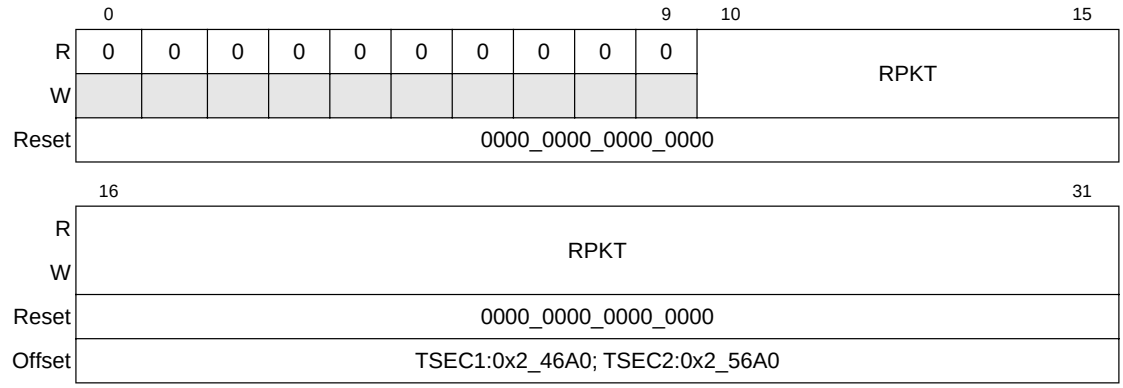


Figure 13-53. Receive Packet Counter Register Definition

Table 13-51 describes the fields of the RPKT register.

Table 13-51. 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).

13.5.3.7.10 Receive FCS Error Counter (RFCS)

Figure 13-54 describes the definition for the RFCS register.

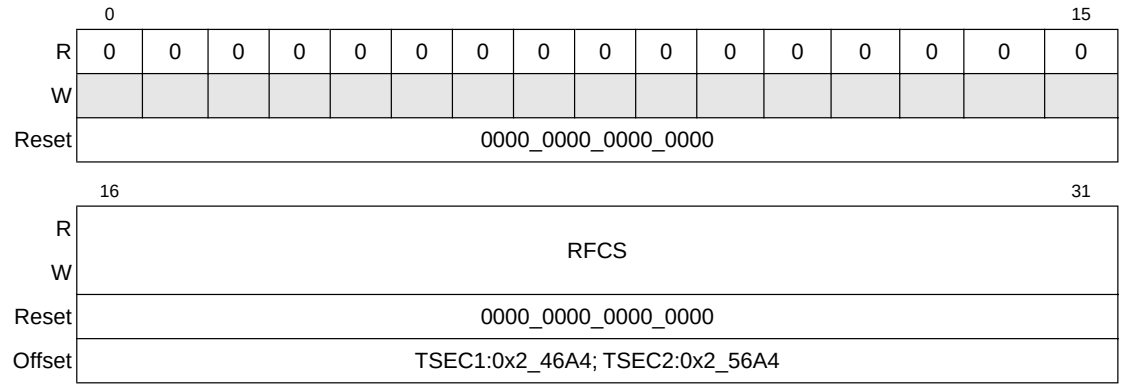


Figure 13-54. Receive FCS Error Counter Register Definition

Table 13-52 describes the fields of the RFCS register.

Table 13-52. 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.

13.5.3.7.11 Receive Multicast Packet Counter (RMCA)

Figure 13-55 describes the definition for the RMCA register.

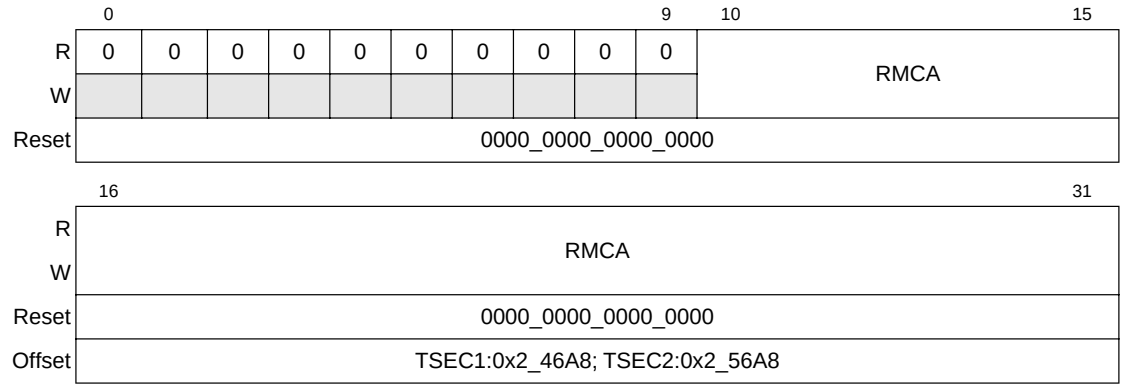


Figure 13-55. Receive Multicast Packet Counter Register Definition

Table 13-53 describes the fields of the RMCA register.

Table 13-53. 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.

13.5.3.7.12 Receive Broadcast Packet Counter (RBCA)

Figure 13-55 describes the definition for the RBCA register.

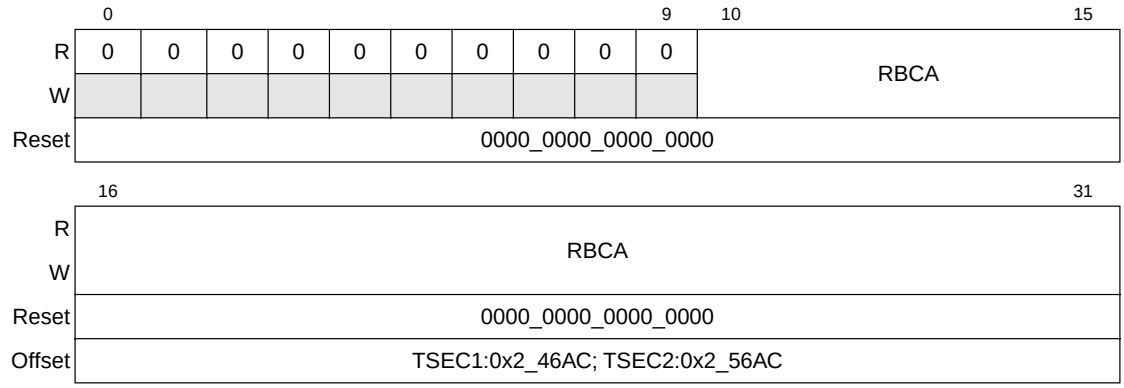


Figure 13-56. Receive Broadcast Packet Counter Register Definition

Table 13-54 describes the fields of the RBCA register.

Table 13-54. 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.

13.5.3.7.13 Receive Control Frame Packet Counter (RXCF)

Figure 13-57 describes the definition for the RXCF register.

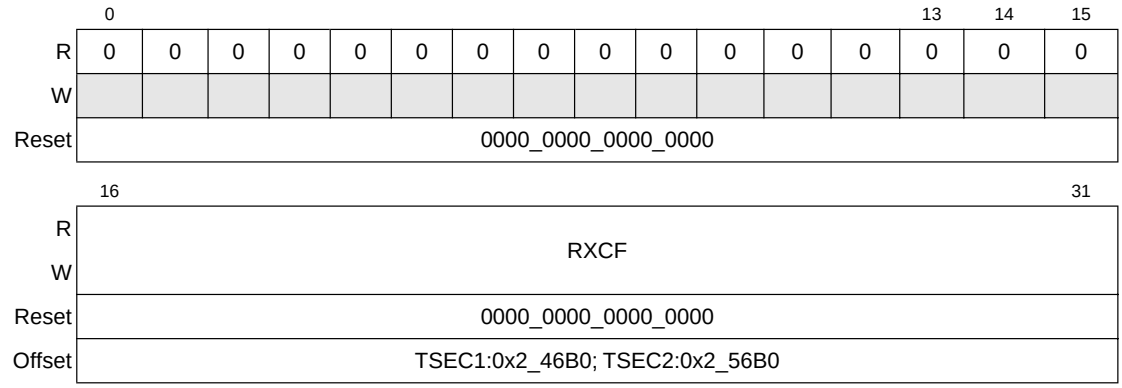


Figure 13-57. Receive Control Frame Packet Counter Register Definition

Table 13-55 describes the fields of the RXCF register.

Table 13-55. 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).

13.5.3.7.14 Receive Pause Frame Packet Counter (RXPF)

Figure 13-58 describes the definition for the RXPF register.

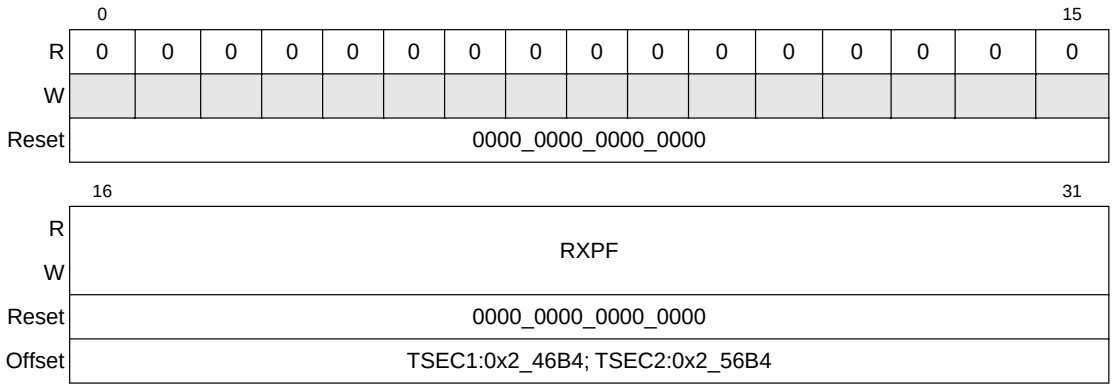


Figure 13-58. Receive Pause Frame Packet Counter Register Definition

Table 13-56 describes the fields of the RXPF register.

Table 13-56. 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.

13.5.3.7.15 Receive Unknown OPCode Packet Counter (RXUO)

Figure 13-59 describes the definition for the RXUO register.

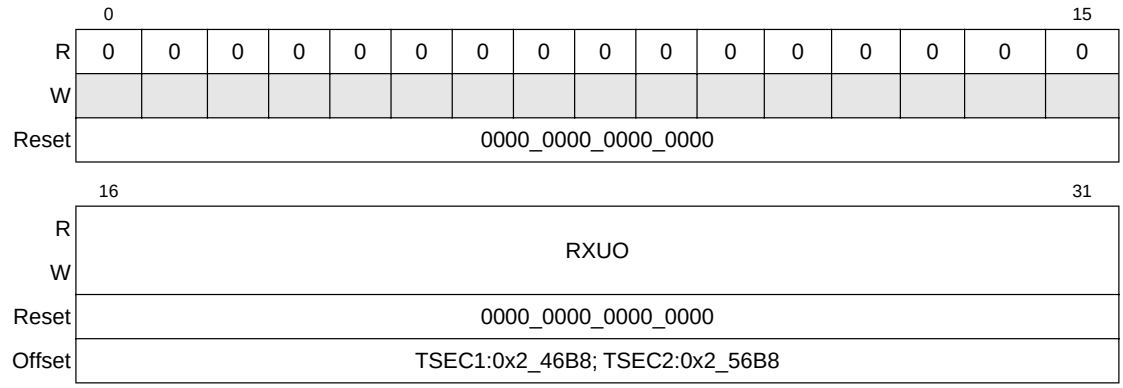


Figure 13-59. Receive Unknown OPCode Packet Counter Register Definition

Table 13-57 describes the fields of the RXUO register.

Table 13-57. 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.

13.5.3.7.16 Receive Alignment Error Counter (RALN)

Figure 13-60 describes the definition for the RALN register.

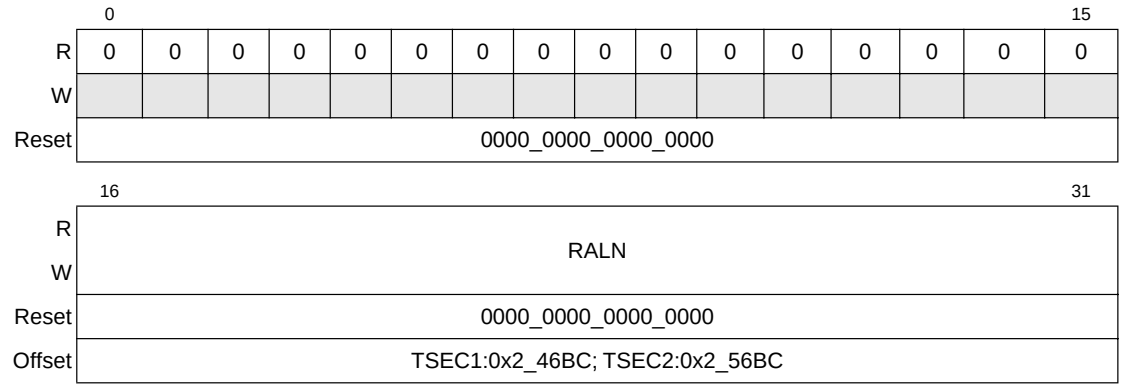


Figure 13-60. Receive Alignment Error Counter Register Definition

Table 13-58 describes the fields of the RALN register.

Table 13-58. 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.

13.5.3.7.17 Receive Frame Length Error Counter (RFLR)

Figure 13-61 describes the definition for the RFLR register.

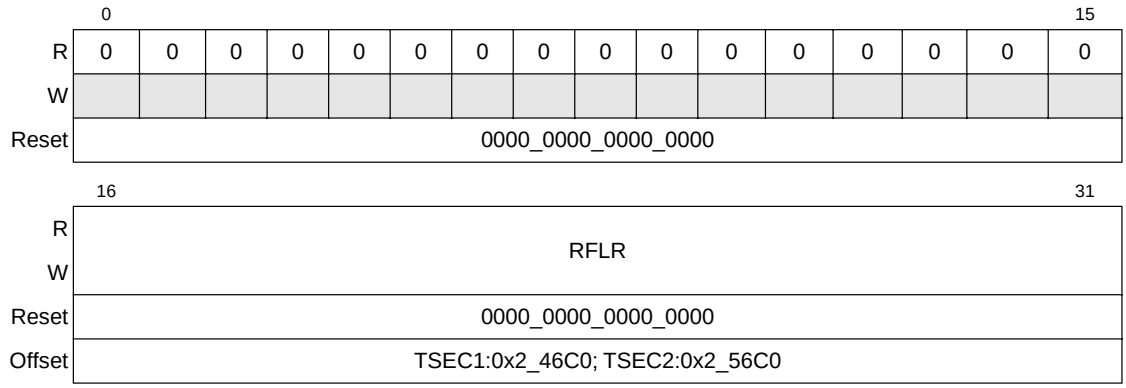


Figure 13-61. Receive Frame Length Error Counter Register Definition

Table 13-59 describes the fields of the RFLR register.

Table 13-59. 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 ethertype value.

13.5.3.7.18 Receive Code Error Counter (RCDE)

Figure 13-62 describes the definition for the RCDE register.

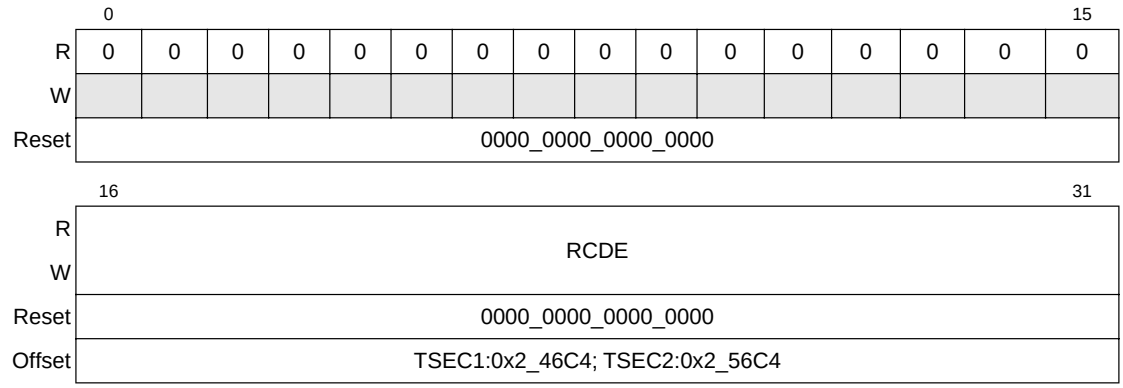


Figure 13-62. Receive Code Error Counter Register Definition

Table 13-60 describes the fields of the RCDE register.

Table 13-60. 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.

13.5.3.7.19 Receive Carrier Sense Error Counter (RCSE)

Figure 13-63 describes the definition for the RCSE register.

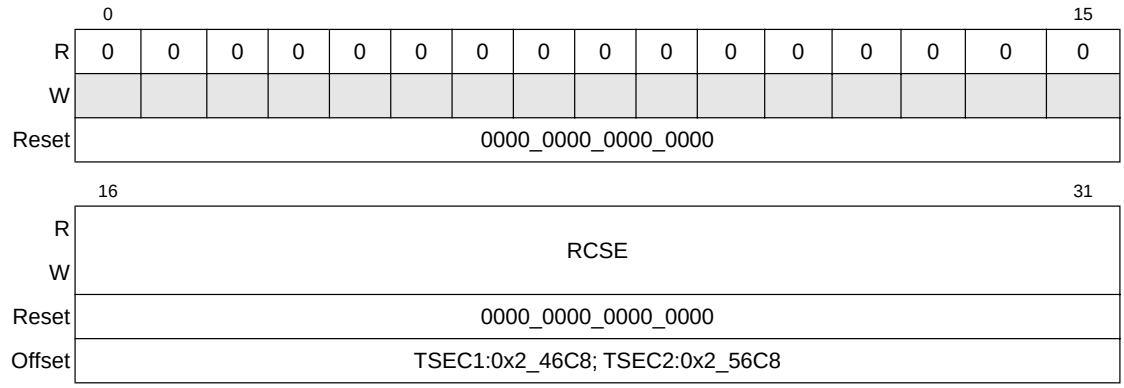


Figure 13-63. Receive Carrier Sense Error Counter Register Definition

Table 13-61 describes the fields of the RCSE register.

Table 13-61. 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 TSEC*n*_RX_ER and an 0xE on TSEC*n*_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.

13.5.3.7.20 Receive Undersize Packet Counter (RUND)

Figure 13-64 describes the definition for the RUND register.

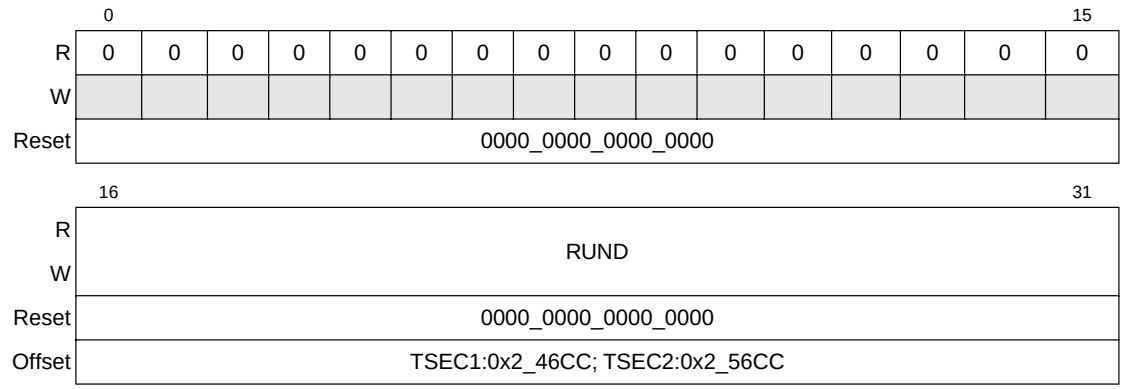


Figure 13-64. Receive Undersize Packet Counter Register Definition

Table 13-62 describes the fields of the RUND register.

Table 13-62. 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.

13.5.3.7.21 Receive Oversize Packet Counter (ROVR)

Figure 13-65 describes the definition for the ROVR register.

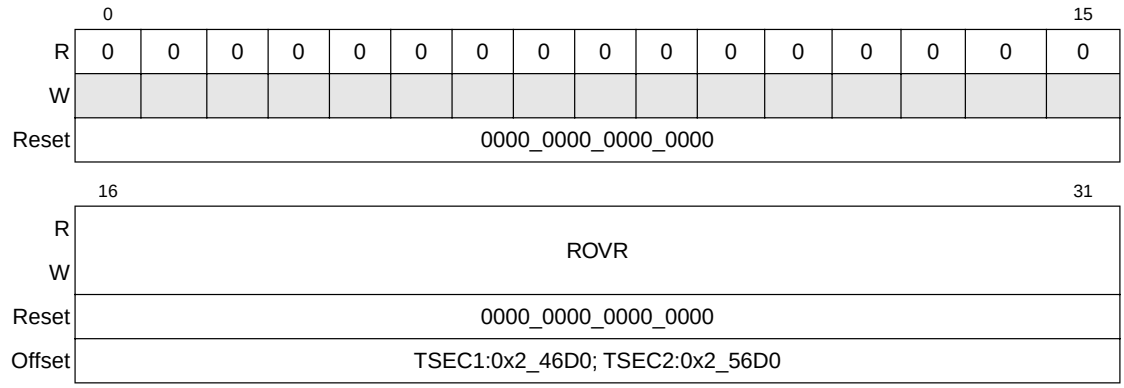


Figure 13-65. Receive Oversize Packet Counter Register Definition

Table 13-63 describes the fields of the ROVR register.

Table 13-63. 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.

13.5.3.7.22 Receive Fragments Counter (RFRG)

Figure 13-66 describes the definition for the RFRG register.

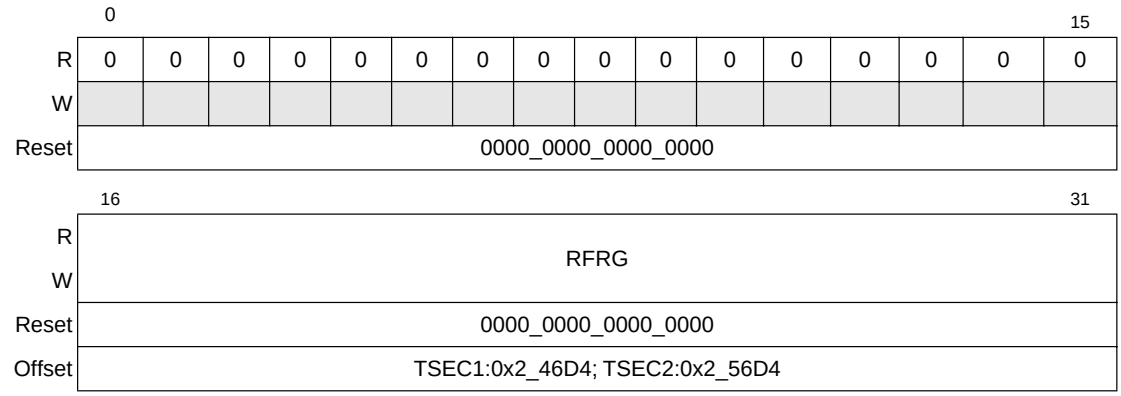


Figure 13-66. Receive Fragments Counter Register Definition

Table 13-64 describes the fields of the RFRG register.

Table 13-64. 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.

13.5.3.7.23 Receive Jabber Counter (RJBR)

Figure 13-67 describes the definition for the RJBR register.

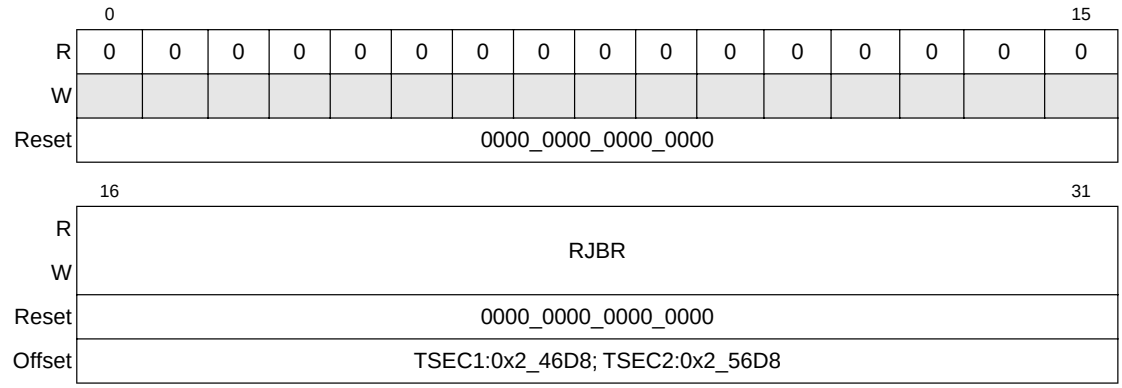


Figure 13-67. Receive Jabber Counter Register Definition

Table 13-65 describes the fields of the RJBR register.

Table 13-65. 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.

13.5.3.7.24 Receive Dropped Packet Counter (RDRP)

Figure 13-68 describes the definition for the RDRP register.

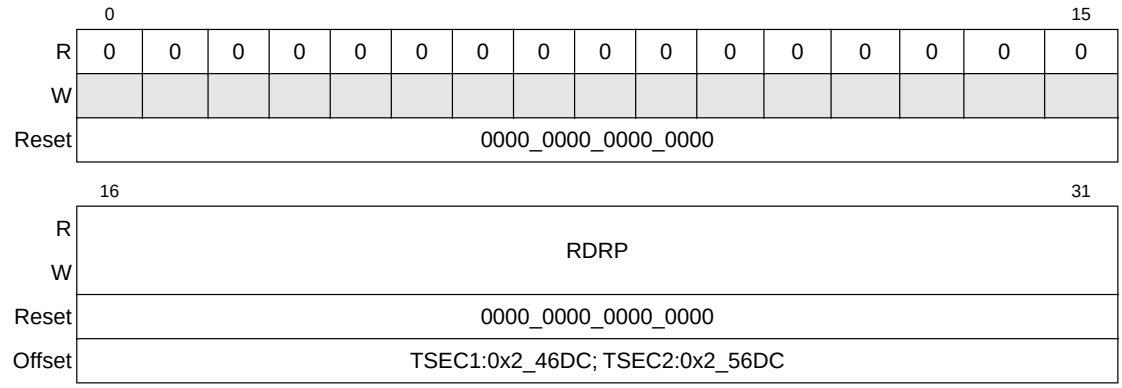


Figure 13-68. Receive Dropped Packet Counter Register Definition

Table 13-66 describes the fields of the RDRP register.

Table 13-66. 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.

13.5.3.7.25 Transmit Byte Counter (TBYT)

Figure 13-69 describes the definition for the TBYT register.

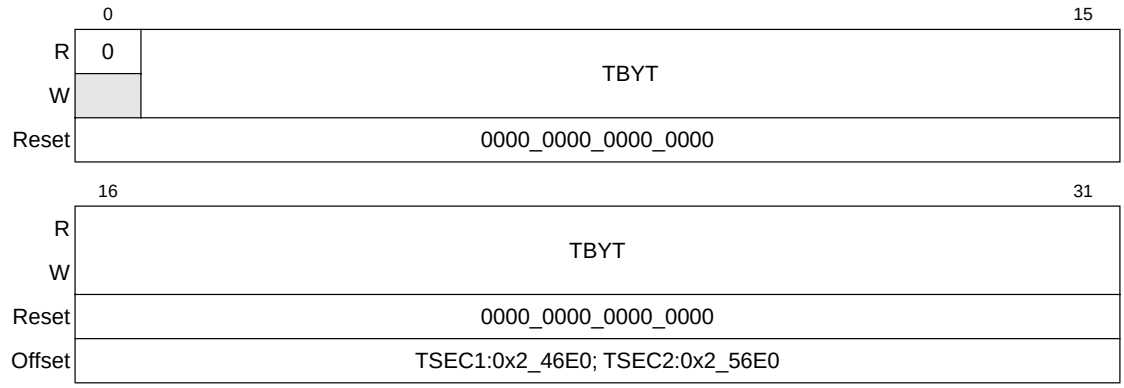


Figure 13-69. Transmit Byte Counter Register Definition

Table 13-67 describes the fields of the TBYT register.

Table 13-67. 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.

13.5.3.7.26 Transmit Packet Counter (TPKT)

Figure 13-70 describes the definition for the TPKT register.

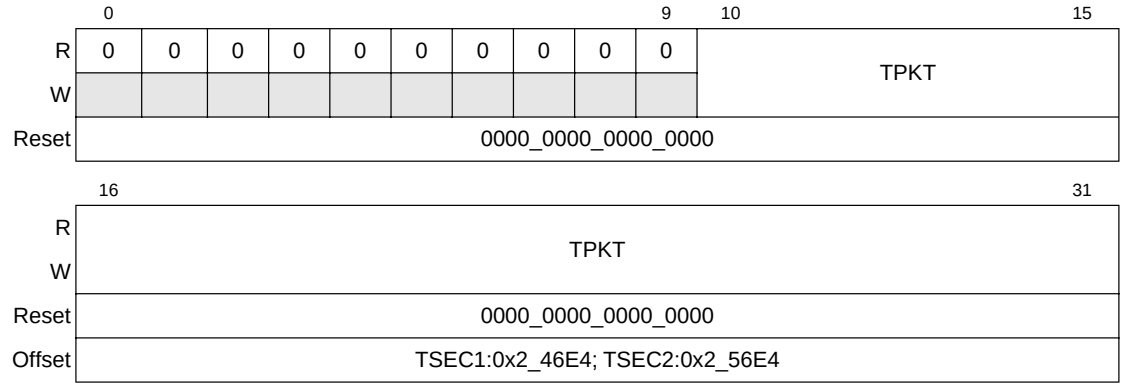


Figure 13-70. Transmit Packet Counter Register Definition

Table 13-68 describes the fields of the TPKT register.

Table 13-68. 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).

13.5.3.7.27 Transmit Multicast Packet Counter (TMCA)

Figure 13-71 describes the definition for the TMCA register.

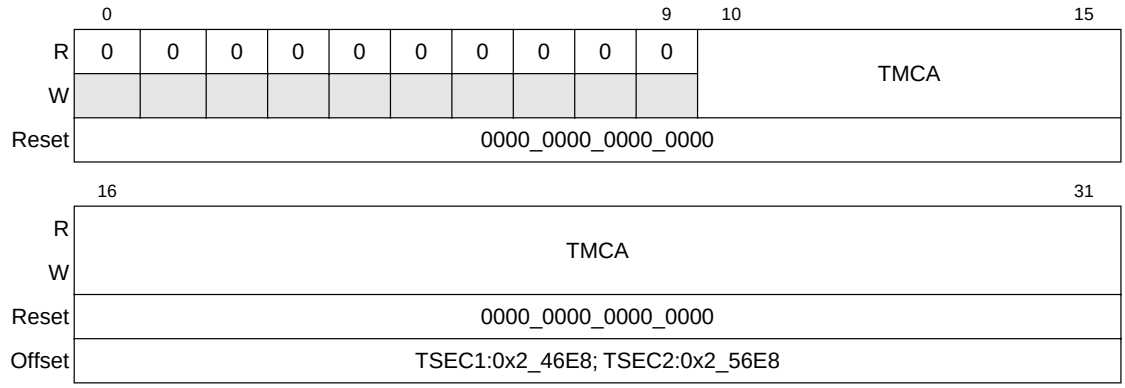


Figure 13-71. Transmit Multicast Packet Counter Register Definition

Table 13-69 describes the fields of the TMCA register.

Table 13-69. 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).

13.5.3.7.28 Transmit Broadcast Packet Counter (TBCA)

Figure 13-72 describes the definition for the TBCA register.

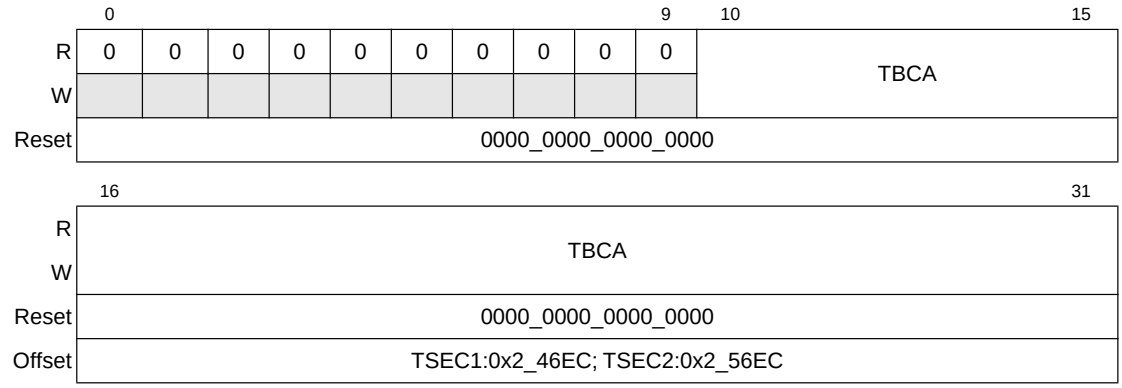


Figure 13-72. Transmit Broadcast Packet Counter Register Definition

Table 13-70 describes the fields of the TBCA register.

Table 13-70. 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).

13.5.3.7.29 Transmit Pause Control Frame Counter (TXPF)

Figure 13-73 describes the definition for the TXPF register.

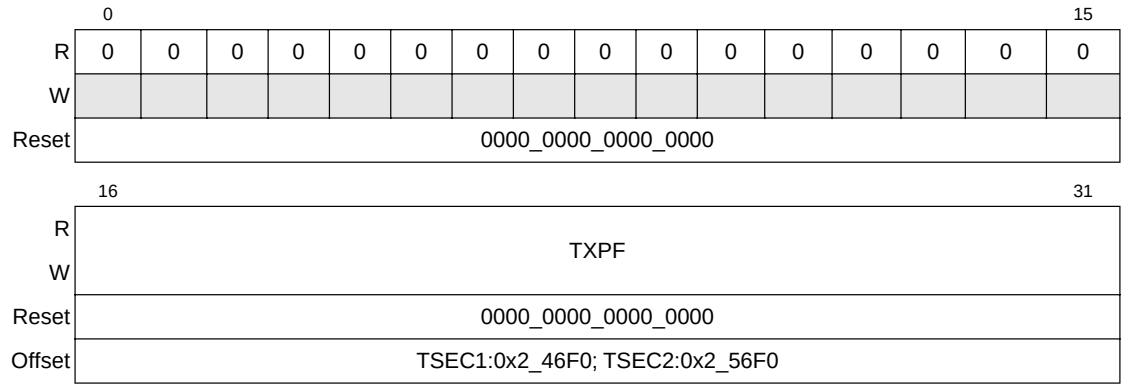


Figure 13-73. Transmit Pause Control Frame Counter Register Definition

Table 13-71 describes the fields of the TXPF register.

Table 13-71. 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.

13.5.3.7.30 Transmit Deferral Packet Counter (TDFR)

Figure 13-74 describes the definition for the TDFR register.

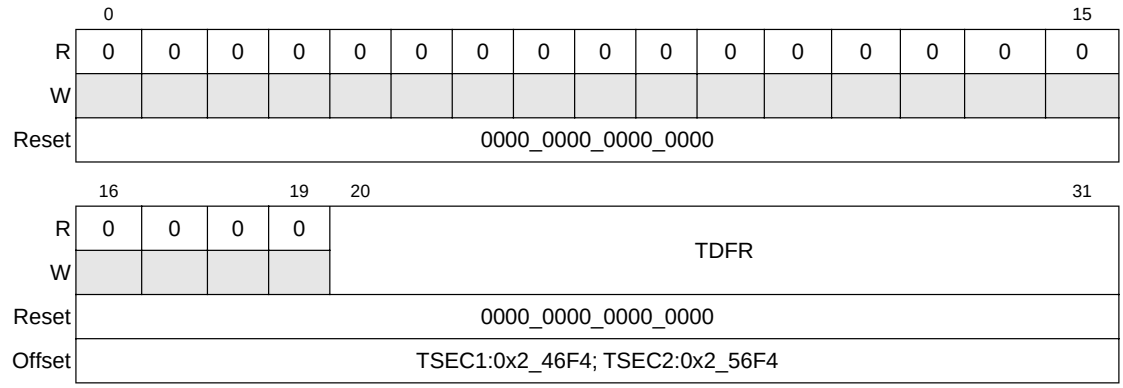


Figure 13-74. Transmit Deferral Packet Counter Register Definition

Table 13-72 describes the fields of the TDFR register.

Table 13-72. 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.

13.5.3.7.31 Transmit Excessive Deferral Packet Counter (TEDF)

Figure 13-75 describes the definition for the TEDF register.

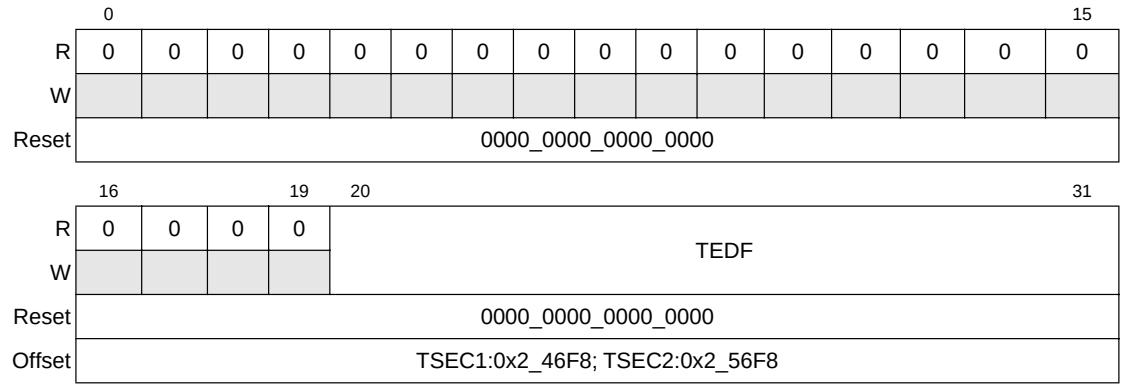


Figure 13-75. Transmit Excessive Deferral Packet Counter Register Definition

Table 13-73 describes the fields of the TEDF register.

Table 13-73. 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).

13.5.3.7.32 Transmit Single Collision Packet Counter (TSCL)

Figure 13-76 describes the definition for the TSCL register.

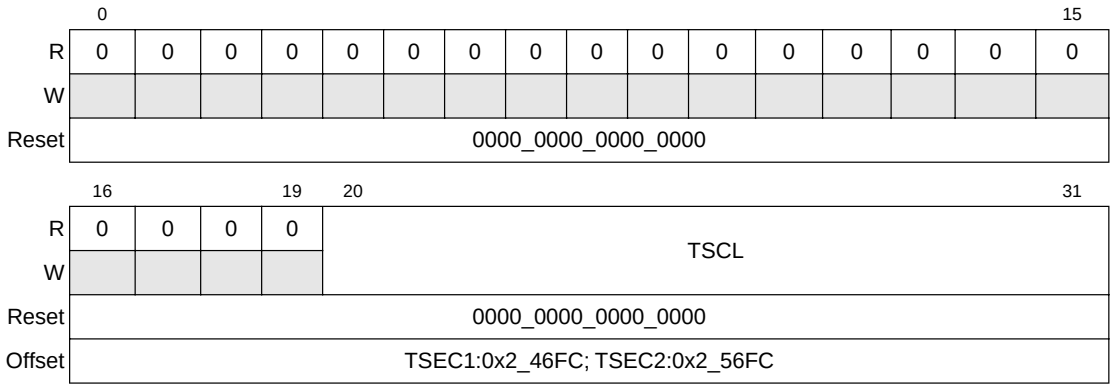


Figure 13-76. Transmit Single Collision Packet Counter Register Definition

Table 13-74 describes the fields of the TSCL register.

Table 13-74. 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.

13.5.3.7.33 Transmit Multiple Collision Packet Counter (TMCL)

Figure 13-77 describes the definition for the TMCL register.

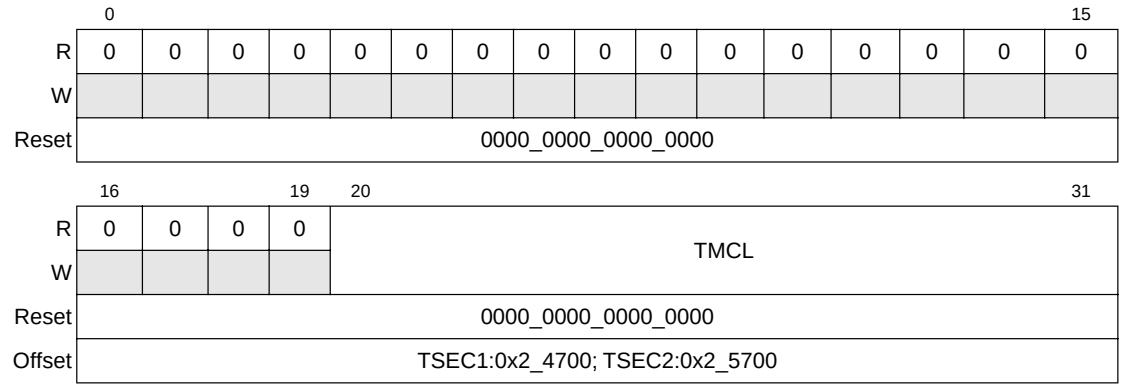


Figure 13-77. Transmit Multiple Collision Packet Counter Register Definition

Table 13-75 describes the fields of the TMCL register.

Table 13-75. 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.

13.5.3.7.34 Transmit Late Collision Packet Counter (TLCL)

Figure 13-78 describes the definition for the TLCL register.

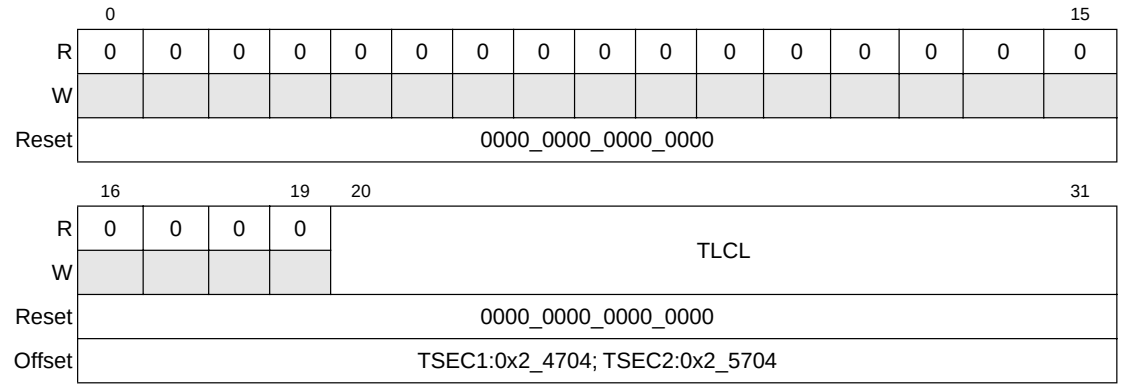


Figure 13-78. Transmit Late Collision Packet Counter Register Definition

Table 13-76 describes the fields of the TLCL register.

Table 13-76. 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.

13.5.3.7.35 Transmit Excessive Collision Packet Counter (TXCL)

Figure 13-79 describes the definition for the TXCL register.

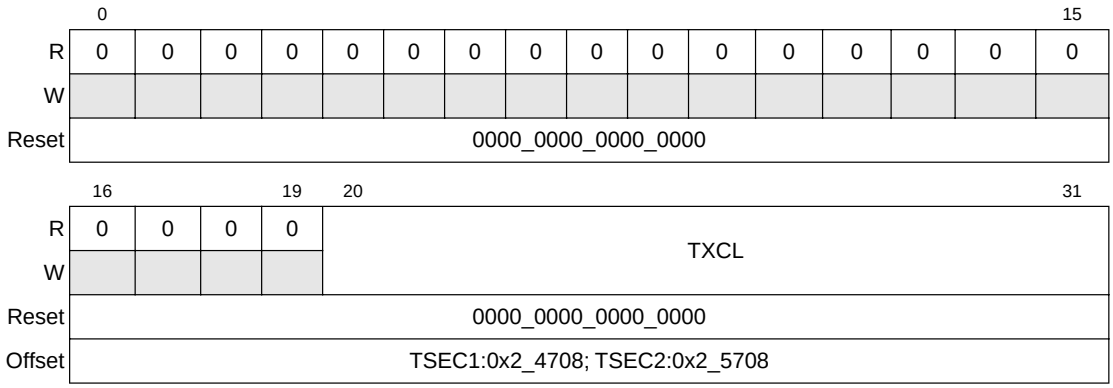


Figure 13-79. Transmit Excessive Collision Packet Counter Register Definition

Table 13-77 describes the fields of the TXCL register.

Table 13-77. 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.

13.5.3.7.36 Transmit Total Collision Counter (TNCL)

Figure 13-80 describes the definition for the TNCL register.

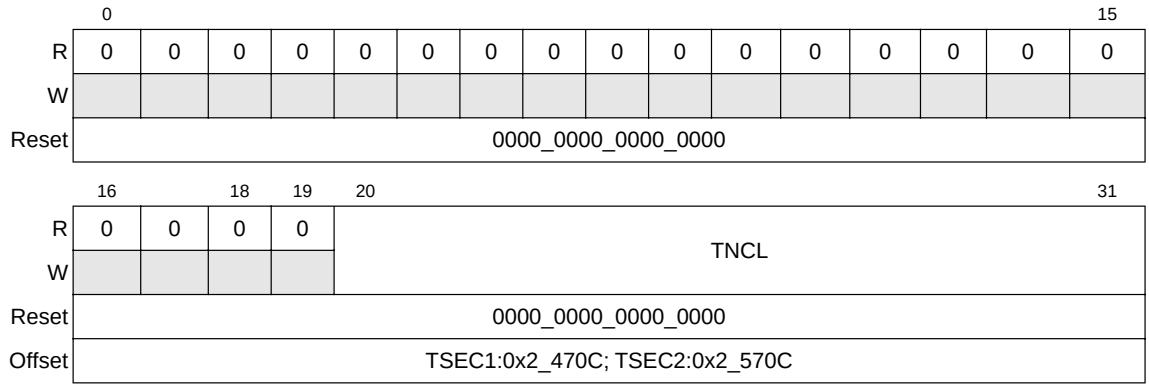


Figure 13-80. Transmit Total Collision Counter Register Definition

Table 13-78 describes the fields of the TNCL register.

Table 13-78. 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.

13.5.3.7.37 Transmit Drop Frame Counter (TDRP)

Figure 13-81 describes the definition for the TDRP register.

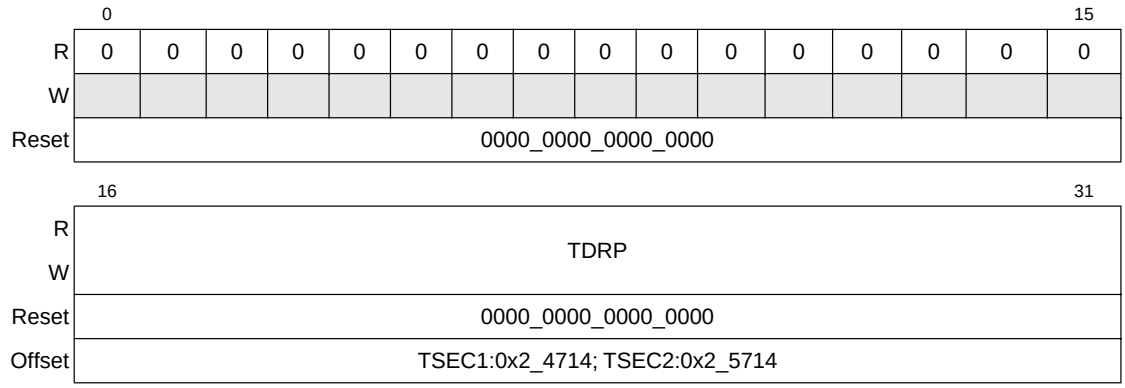


Figure 13-81. Transmit Drop Frame Counter Register Definition

Table 13-79 describes the fields of the TDRP register.

Table 13-79. TDRP Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TDRP	Transmit drop frame counter. Increments each time input PFH is asserted.

13.5.3.7.38 Transmit Jabber Frame Counter (TJBR)

Figure 13-82 describes the definition for the TJBR register.

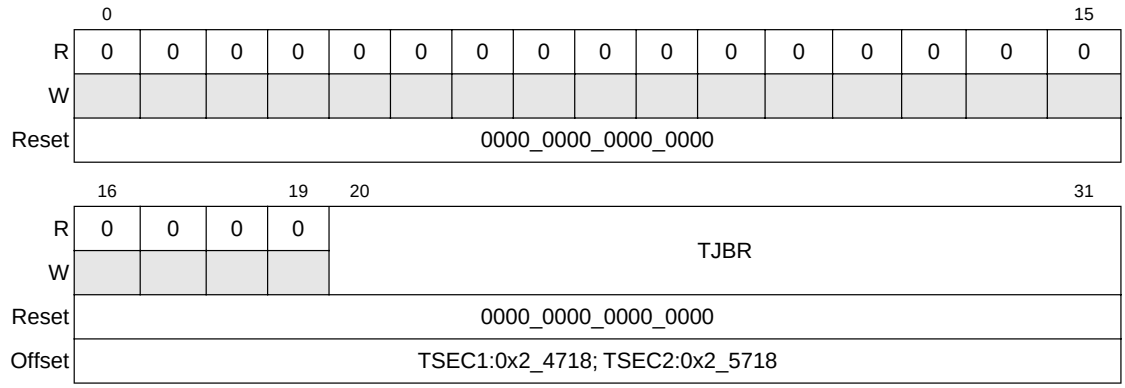


Figure 13-82. Transmit Jabber Frame Counter Register Definition

Table 13-80 describes the fields of the TJBR register.

Table 13-80. 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.

13.5.3.7.39 Transmit FCS Error Counter (TFCS)

Figure 13-83 describes the definition for the TFCS register.

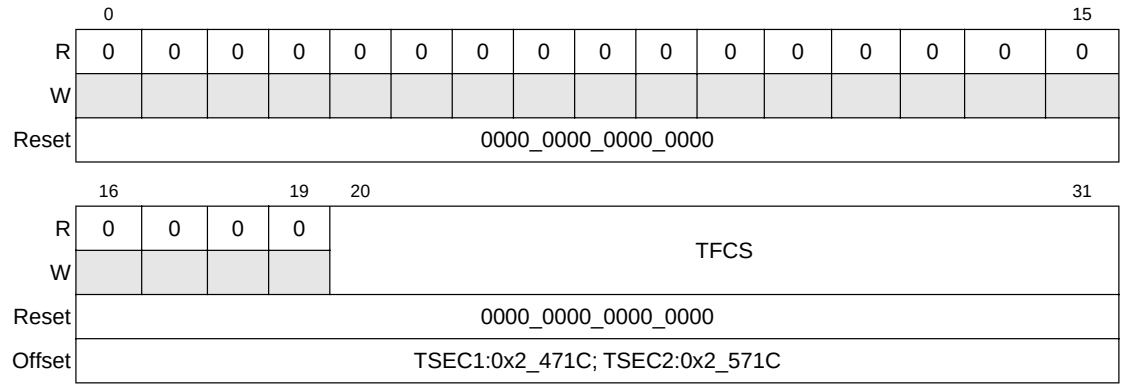


Figure 13-83. Transmit FCS Error Counter Register Definition

Table 13-81 describes the fields of the TFCS register.

Table 13-81. 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.

13.5.3.7.40 Transmit Control Frame Counter (TXCF)

Figure 13-84 describes the definition for the TXCF register.

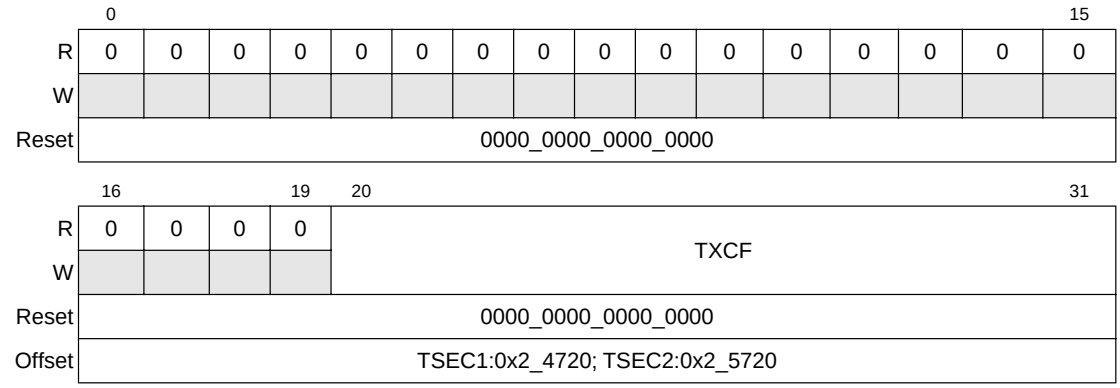


Figure 13-84. Transmit Control Frame Counter Register Definition

Table 13-82 describes the fields of the TXCF register.

Table 13-82. 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.

13.5.3.7.41 Transmit Oversize Frame Counter (TOVR)

Figure 13-85 describes the definition for the TOVR register.

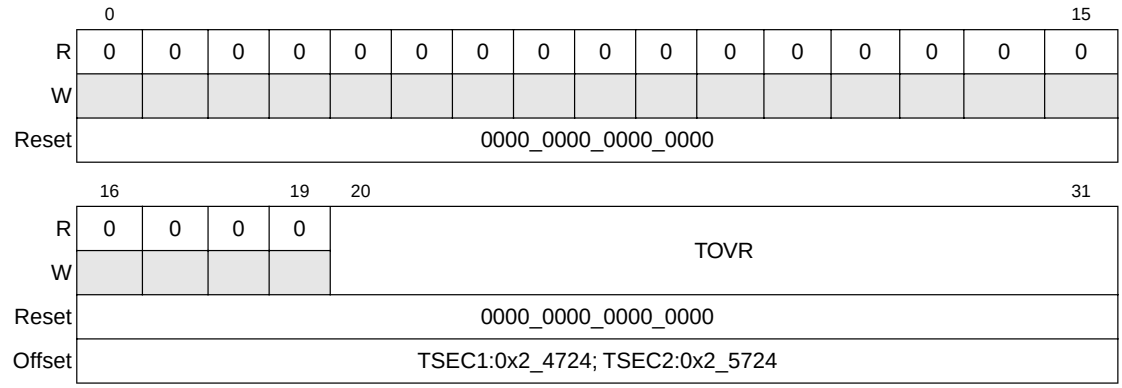


Figure 13-85. Transmit Oversized Frame Counter Register Definition

Table 13-83 describes the fields of the TOVR register.

Table 13-83. 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.

13.5.3.7.42 Transmit Undersize Frame Counter (TUND)

Figure 13-86 describes the definition for the TUND register.

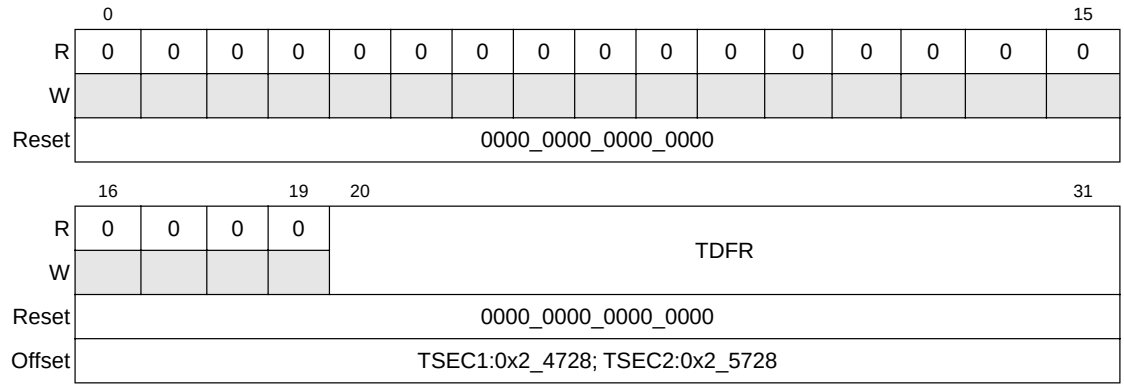


Figure 13-86. Transmit Undersize Frame Counter Register Definition

Table 13-84 describes the fields of the TUND register.

Table 13-84. 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.

13.5.3.7.43 Transmit Fragment Counter (TFRG)

Figure 13-87 describes the definition for the TFRG register.

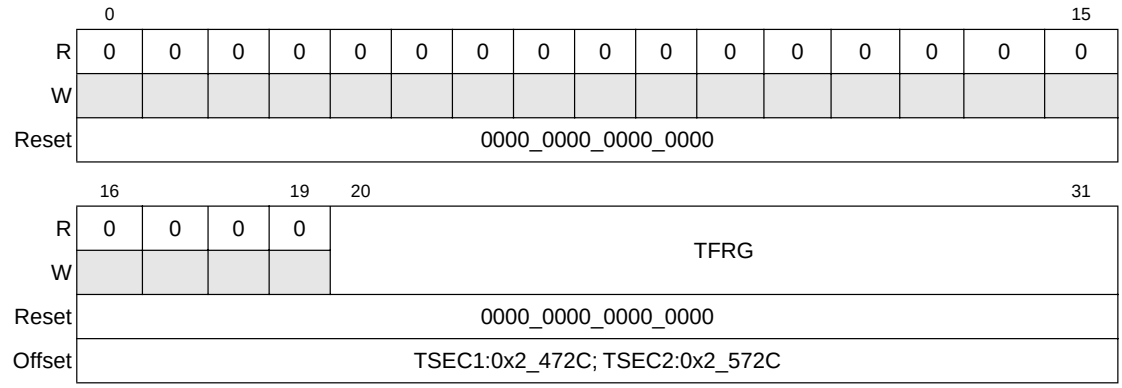


Figure 13-87. Transmit Fragment Counter Register Definition

Table 13-85 describes the fields of the TFRG register.

Table 13-85. TFRG Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TFRG	Transmit fragment counter. Increments for every frame less then 64 bytes, with an incorrect FCS value.

13.5.3.7.44 Carry Register One (CAR1)

Carry register bits are cleared on carry register writes when the respective bits are set. Figure 13-88 describes the definition for the CAR1 register.

	0	1	2	3	4	5	6	7							14	15
R	C1 64	C1 127	C1 255	C1 511	C1 1K	C1 MAX	C1 MGV	0	0	0	0	0	0	0	0	C1 RBY
W																
Reset	0000_0000_0000_0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	C1 RPK	C1 RFC	C1 RMC	C1 RBC	C1 RXC	C1 RXP	C1 RXU	C1 RAL	C1 RFL	C1 RCD	C1 RCS	C1 RUN	C1 ROV	C1 RFR	C1 RJB	C1 RDR
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4730; TSEC2:0x2_5730															

Figure 13-88. Carry Register One (CAR1) Register Definition

Table 13-86 describes the fields of the CAR1 register.

Table 13-86. 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 13-86. 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

13.5.3.7.45 Carry Register Two (CAR2)

Figure 13-89 describes the definition for the CAR2 register.

	0										11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	C2 TJB	C2 TFC	C2 TCF	C2 TOV
W															
Reset	0000_0000_0000_0000														

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	C2 TUN	C2 TFG	C2 TBY	C2 TPK	C2 TMC	C2 TBC	C2 TPF	C2 TDF	C2 TED	C2 TSC	C2 TMA	C2 TLC	C2 TXC	C2 TNC	C2 TPH	C2 TDP
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4734; TSEC2:0x2_5734															

Figure 13-89. Carry Register Two (CAR2) Register Definition

Carry register bits are cleared on carry register write when the respective bits are set. Table 13-87 describes the fields of the CAR2 register.

Table 13-87. 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 13-87. 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

13.5.3.7.46 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 13-90 describes the definition for the CAM1 register.

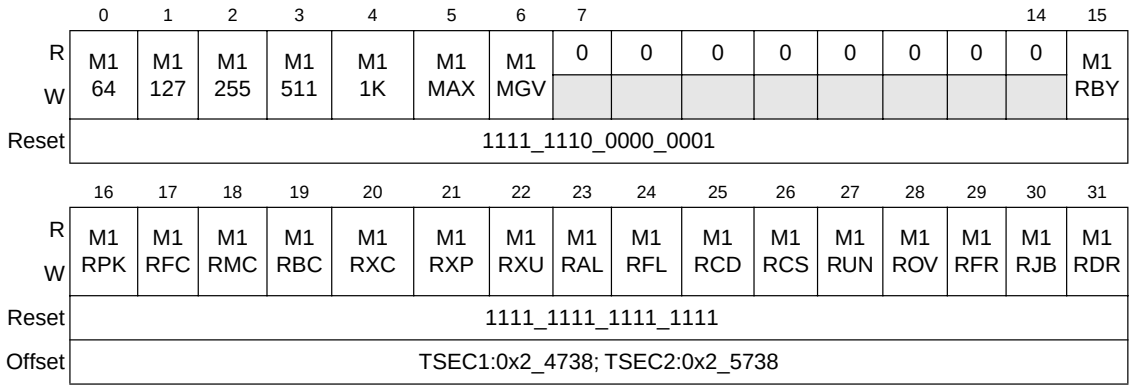


Figure 13-90. Carry Mask Register One (CAM1) Register Definition

Table 13-88 describes the fields of the CAM1 register.

Table 13-88. 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

13.5.3.7.47 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 13-91 describes the definition for the CAM2 register.

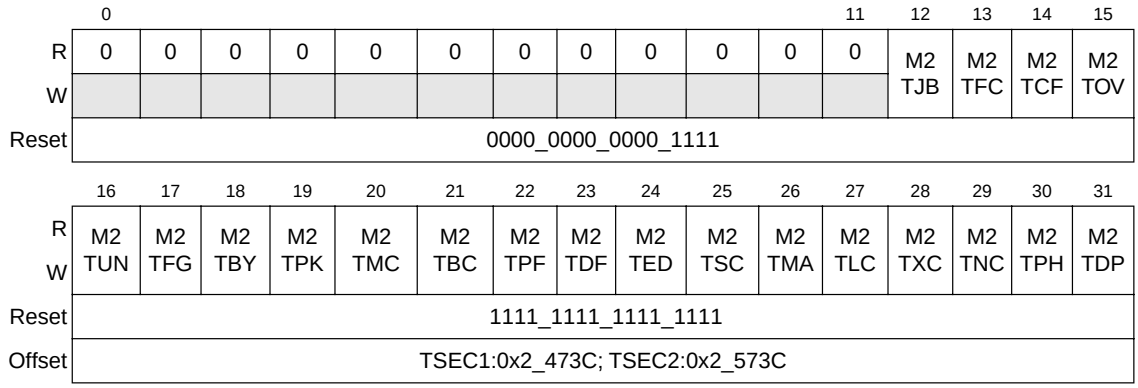


Figure 13-91. Carry Mask Register Two (CAM2) Register Definition

Table 13-89 describes the fields of the CAM2 register.

Table 13-89. 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

13.5.3.8 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 13.6.2.6.2, “Hash Table Algorithm” for more information on the hash algorithm.

13.5.3.8.1 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. 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 13-92 describes the definition for the IADDR*n* register.

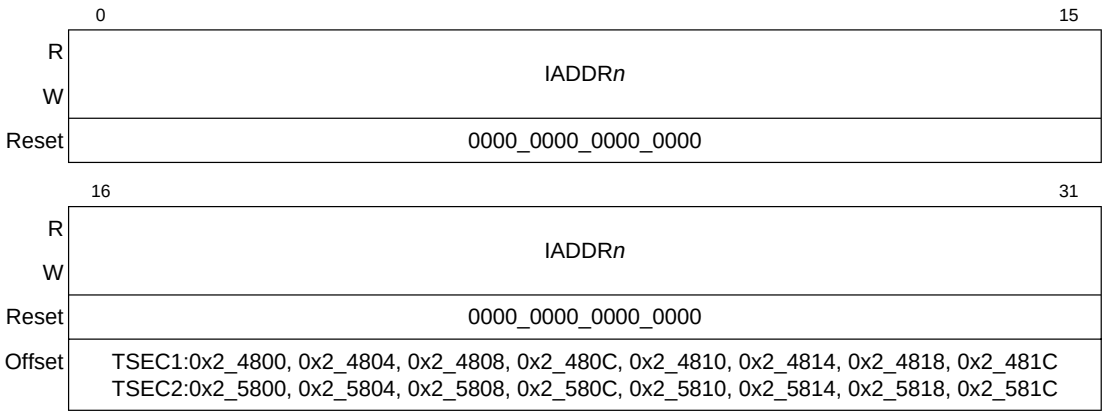


Figure 13-92. IADDR*n* Register Definition

Table 13-90 describes the fields of the IADDR*n* register.

Table 13-90. IADDR*n* Field Description

Bits	Name	Description
0–31	IADDR*n*	Represents the 32-bit value associated with the corresponding register. IADDR0 contains the high-order 32 bits of the 256-entry hash table and IADDR7 represents the low-order 32 bits.

13.5.3.8.2 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. 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 13-93 describes the definition for the GADDR*n* register.

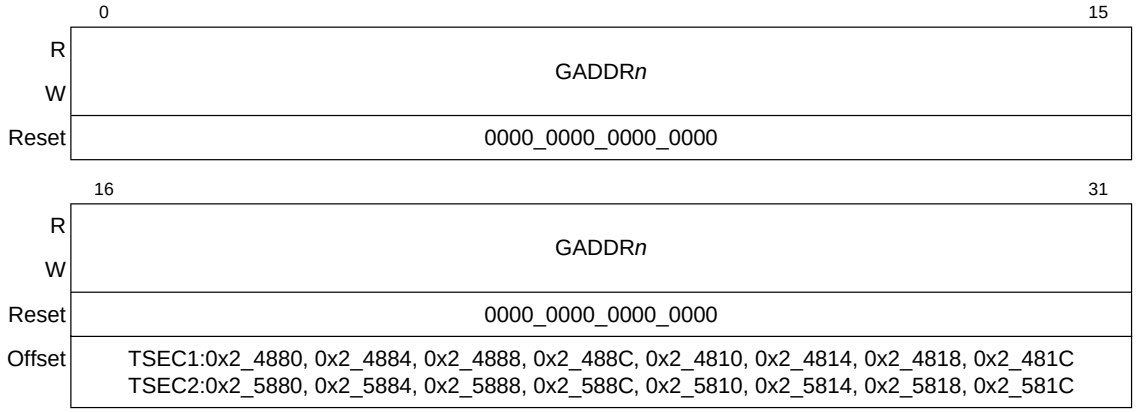


Figure 13-93. GADDR*n* Register Definition

Table 13-91 describes the fields of the GADDR*n* register.

Table 13-91. 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.

13.5.3.8.3 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 13-94 describes the definition for the ATTR register.

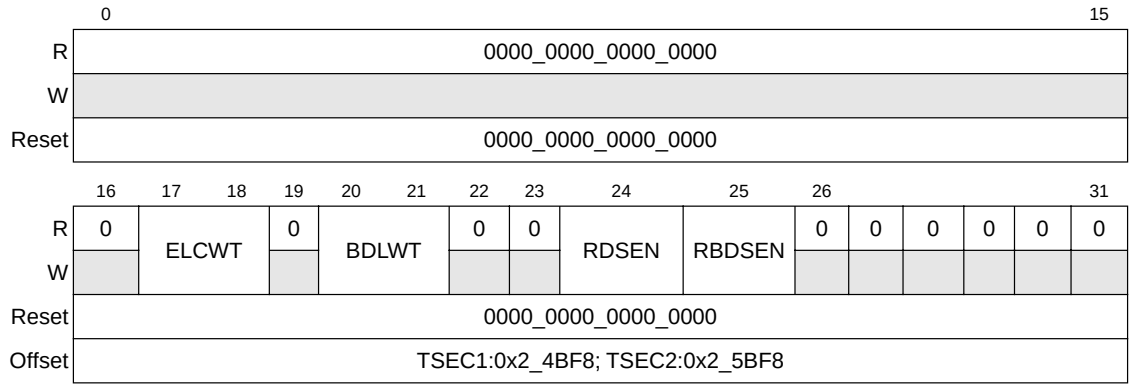


Figure 13-94. ATTR Register Definition

Table 13-92 describes the fields of the ATTR register.

Table 13-92. 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

13.5.3.8.4 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 13-95 describes the definition for the ATTRELI register.

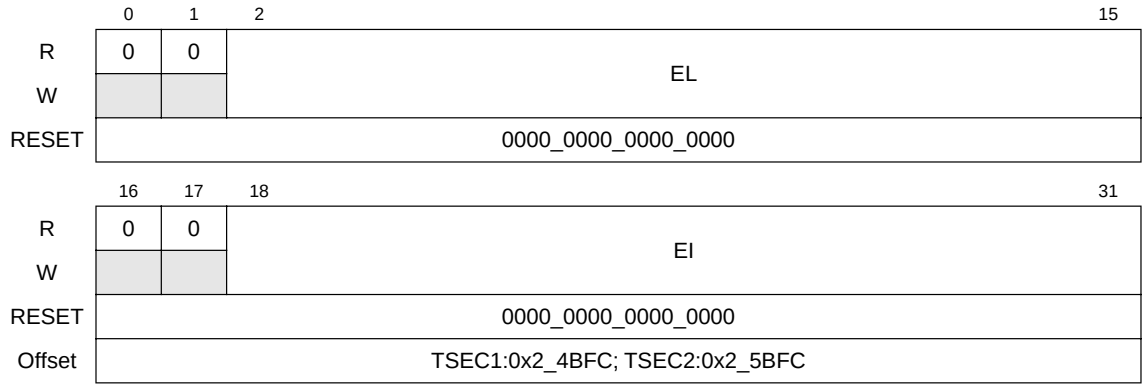


Figure 13-95. ATTRELI Register Definition

Table 13-93 describes the fields of the ATTRELI register.

Table 13-93. 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.

13.5.4 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) and the TBI MII set of registers.

13.5.4.1 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 13-94), 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 13.5.3.1, "TSEC General Control and Status Registers," and the TBI MII register set in Table 13-94. 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 13-94. TBI MII Register Set

Offset Address	Name	Access	Size	Section/Page
TEN-BIT INTERFACE (TBI) REGISTERS				13.5.4/13-98
0x00	Control (CR)	R/W [1]	16 bits	13.5.4.2/13-99
0x01	Status (SR)	R, LH, LL	16 bits	13.5.4.3/13-100
0x02–0x03	Reserved	R	2 bytes	—
0x04	AN advertisement (ANA)	RW, R	16 bits	13.5.4.3/13-100
0x05	AN link partner base page ability (ANLPBPA)	R	16 bits	13.5.4.5/13-103
0x06	AN expansion (ANEX)	R, LH	16 bits	13.5.4.6/13-104
0x07	AN next page transmit (ANNPT)	R/W, R	16 bits	13.5.4.7/13-105
0x08	AN link partner ability next page (ANLPANP)	R	16 bits	13.5.4.8/13-106
0x0F	Extended status (EXST)	R	16 bits	13.5.4.9/13-107
0x10	Jitter diagnostics (JD)	R/W	16 bits	13.5.4.10/13-108
0x11	TBI control (TBICON)	R/W	16 bits	13.5.4.11/13-109

[1] R = means Read-only, WO = Write Only, R/W = Read and Write, LH = Latches High, LL = Latches Low, SC = Self-clearing,

13.5.4.2 Control Register (CR)

Figure 13-96 describes the definition for the CR register.

	0	1	2	3	4	5	6	7	8	9	10					15
R	PHY Reset	Loop Back	Speed [0]	AN Enable	0	0	Reset AN	Full Duplex	0	Speed [1]	0	0	0	0	0	0
W																
Reset	0001_0001_0100_0000															
Offset	0x00															

Figure 13-96. Control Register Definition

Table 13-95 describes the fields of the CR register.

Table 13-95. CR Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>PHY Reset</td><td>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.</td></tr>
<tr><td>1</td><td>Loop Back</td><td>Loop back. This bit is cleared by default.
0 Normal operation.
1 The transmit outputs of the TBI are connected to the receive inputs.</td></tr>
<tr><td>2</td><td>Speed[0]</td><td>Speed selection. This bit defaults to a cleared state and should always be cleared, which corresponds to 1000 Mbps speed.</td></tr>
<tr><td>3</td><td>AN Enable</td><td>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.</td></tr>
<tr><td>4–5</td><td>—</td><td>Reserved</td></tr>
<tr><td>6</td><td>Reset AN</td><td>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.</td></tr>
<tr><td>7</td><td>Full Duplex</td><td>Duplex mode. This bit is set by default.
0 Half-duplex operation.
1 Full-duplex operation.</td></tr>
<tr><td>8</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>9</td><td>Speed[1]</td><td>Speed selection. This bit defaults to a set state and should always be set, which corresponds to 1000 Mbps speed.
<table>
<tr><th>Maximum Operating Speed</th><th>Bit 2</th><th>Bit 9</th></tr>
<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>
</table>
</td></tr>
<tr><td>10–15</td><td>—</td><td>Reserved</td></tr>
</table>

13.5.4.3 Status Register (SR)

Figure 13-97 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	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 13-97. Status Register Definition

Table 13-96 describes the fields of the SR register.

Table 13-96. 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.

13.5.4.4 AN Advertisement Register (ANA)

Figure 13-98 describes the definition for the ANA 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	0x04													

Figure 13-98. AN Advertisement Register Definition

Table 13-97 describes the fields of the ANA register.

Table 13-97. 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. **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 13-98. **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 13-97. 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 13-98 describes the resolution of pause priority.

Table 13-98. 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

13.5.4.5 AN Link Partner Base Page Ability Register (ANLPBPA)

Figure 13-99 describes the definition for the ANLPBPA register.

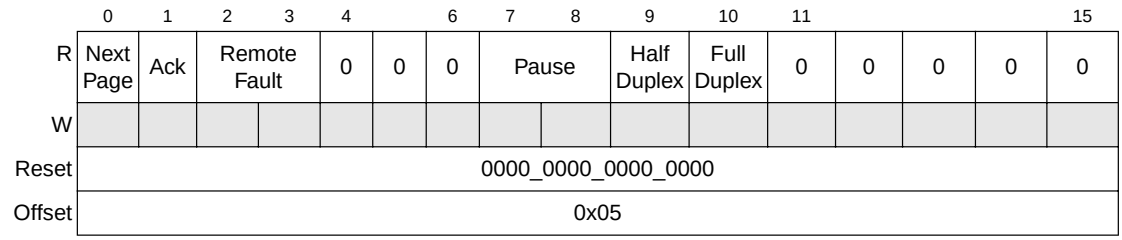


Figure 13-99. AN Link Partner Base Page Ability Register Definition

Table 13-99 describes the fields of the ANLPBPA register.

Table 13-99. ANLPBPA Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>Next Page</td><td>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.</td></tr>
<tr><td>1</td><td>Ack</td><td>Acknowledge. Ignore on read. This bit is read-only</td></tr>
<tr><td>2–3</td><td>Remote Fault</td><td>The link partner’s remote fault condition is encoded in bits 2 and 3 of the base page. Values are shown in the remote fault encoding field table below. This bit is read-only.
<table>
<tr><th>RF1 bit[3]</th><th>RF2 bit[2]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>No error, link OK</td></tr>
<tr><td>0</td><td>1</td><td>Offline</td></tr>
<tr><td>1</td><td>0</td><td>Link_Failure</td></tr>
<tr><td>1</td><td>1</td><td>Auto-Negotiation_Error</td></tr>
</table></td></tr>
<tr><td>4–6</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>7–8</td><td>Pause</td><td>Encoding of the link partner’s PAUSE capability is shown in the PAUSE encoding table below. For priority resolution information consult. This bit is read-only
<table>
<tr><th>PAUSE bit[8]</th><th>ASM_DIR bit[7]</th><th>Capability</th></tr>
<tr><td>0</td><td>0</td><td>No PAUSE</td></tr>
<tr><td>0</td><td>1</td><td>Asymmetric PAUSE toward link partner</td></tr>
<tr><td>1</td><td>0</td><td>Symmetric PAUSE</td></tr>
<tr><td>1</td><td>1</td><td>Both symmetric PAUSE and
Asymmetric PAUSE toward local device</td></tr>
</table></td></tr>
<tr><td>9</td><td>Half Duplex</td><td>Half-duplex capability. This bit is read-only.
0 Link partner is not capable of half-duplex mode.
1 Link partner is capable of half-duplex mode.</td></tr>
<tr><td>10</td><td>Full Duplex</td><td>Full-duplex capability. This bit is read-only.
0 Link partner is not capable of full-duplex mode.
1 Link partner is capable of full-duplex mode.</td></tr>
<tr><td>11–15</td><td>—</td><td>Reserved, should be cleared.</td></tr>
</table>

13.5.4.6 AN Expansion Register (ANEX)

Figure 13-100 describes the definition for the ANEX register.

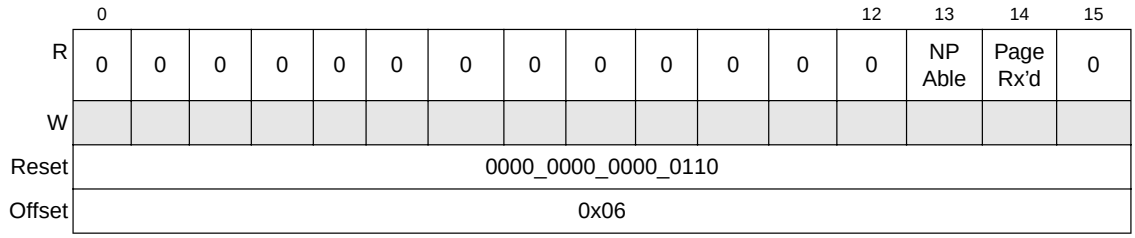


Figure 13-100. AN Expansion Register Definition

Table 13-100 describes the fields of the ANEX register.

Table 13-100. 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.

13.5.4.7 AN Next Page Transmit Register (ANNPT)

Figure 13-101 describes the definition for the ANNPT register.

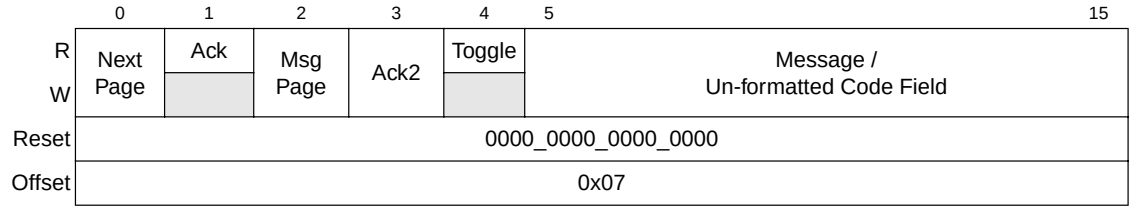


Figure 13-101. AN Next Page Transmit Register Definition

Table 13-101 describes the fields of the ANNPT register.

Table 13-101. 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]

13.5.4.8 AN Link Partner Ability Next Page Register (ANLPANP)

Figure 13-102 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 13-102. AN Link Partner Ability Next Page Register Definition

Table 13-102 describes the fields of the ANLPANP register.

Table 13-102. 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.

13.5.4.9 Extended Status Register (EXST)

Figure 13-103 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	0
W																
Reset	1010_0000_0000_0000															
Offset	0x0F															

Figure 13-103. Extended Status Register Definition

Table 13-103 describes the fields of the EXST register.

Table 13-103. 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

13.5.4.10 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. In may be wise to auto-negotiate and advertise a remote fault signaling of offline prior to beginning the test patterns. Figure 13-104 describes the definition for the JD register.

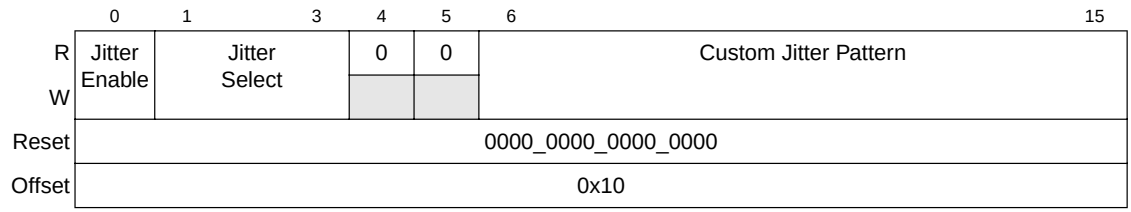


Figure 13-104. Jitter Diagnostics Register Definition

Table 13-104 describes the fields of the JD register.

Table 13-104. 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.

Jitter Pattern Select	bit[1]	bit[2]	bit[3]
User defined uses custom jitter pattern, bits 6–15	0	0	0
High frequency (+/- D21.5) 1010101010101010101010101010101010101010…	0	0	1
Mixed frequency (+/- K28.5) 1111101011000001010011111010110000010100…	0	1	0
Low frequency 1111100000111110000011111000001111100000…	0	1	1
Complex pattern (10’h17c,10’h0c9,10’h0e5,10’h2a3, 10’h17c,...)	1	0	0
Square Wave (- K28.7) 0011111000001111100000111110000011111000…	1	0	1
Reserved	1	1	0
Reserved	1	1	1

4–5	—	Reserved													
6–15	Custom Jitter Pattern	Used in conjunction with jitter (pattern) select and jitter (diagnostic) enable; set this field to the desired custom pattern which is continuously transmitted. Its default is 0x000.													

13.5.4.11 TBI Control Register (TBICON)

Figure 13-105 describes the definition for the TBICON register.

	0	1	2	3	4	5	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 13-105. TBI Control Register Definition

Table 13-105 describes the fields of the TBICON register.

Table 13-105. 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

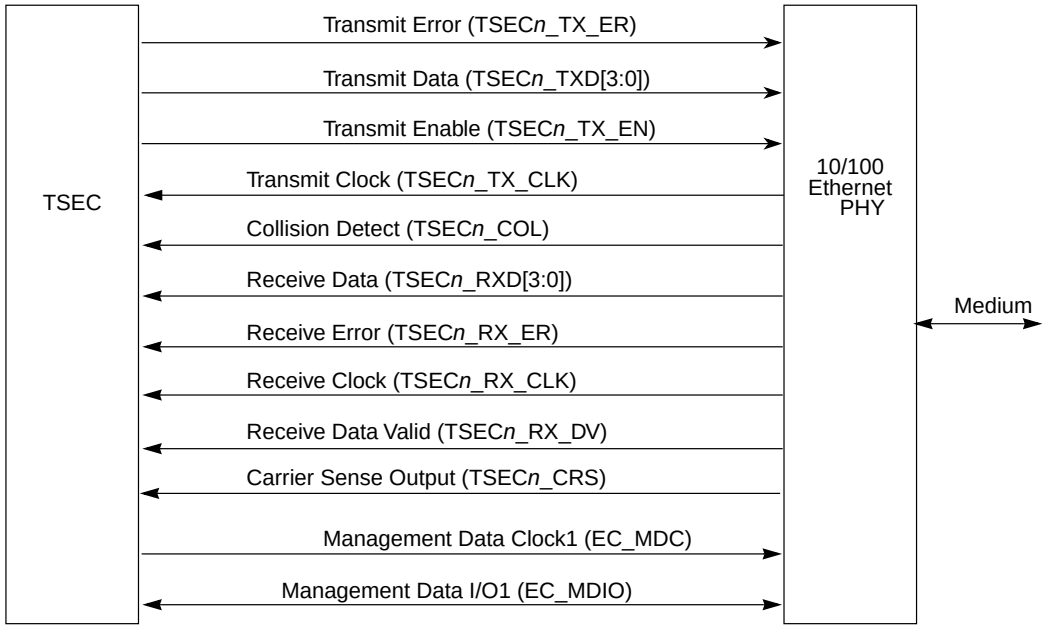
13.6 Functional Description

13.6.1 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).

13.6.1.1 Media-Independent Interface (MII)

This section describes the media-independent interface (MII) intended to be used between the PHYs and the TSEC. Figure 13-106 depicts the basic components of the MII including the signals required to establish TSEC module connection with a PHY.



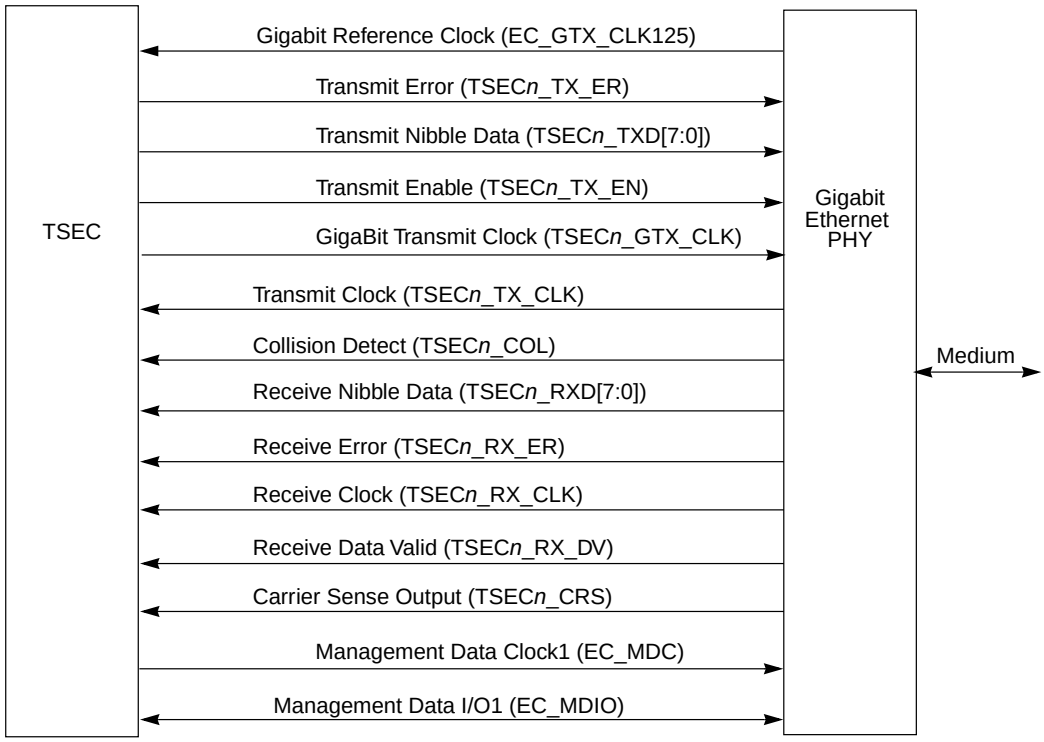
[1] The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 13-106. 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.

13.6.1.2 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 13-107 depicts the basic components of the GMII including the signals required to establish the TSEC module connection with a PHY.



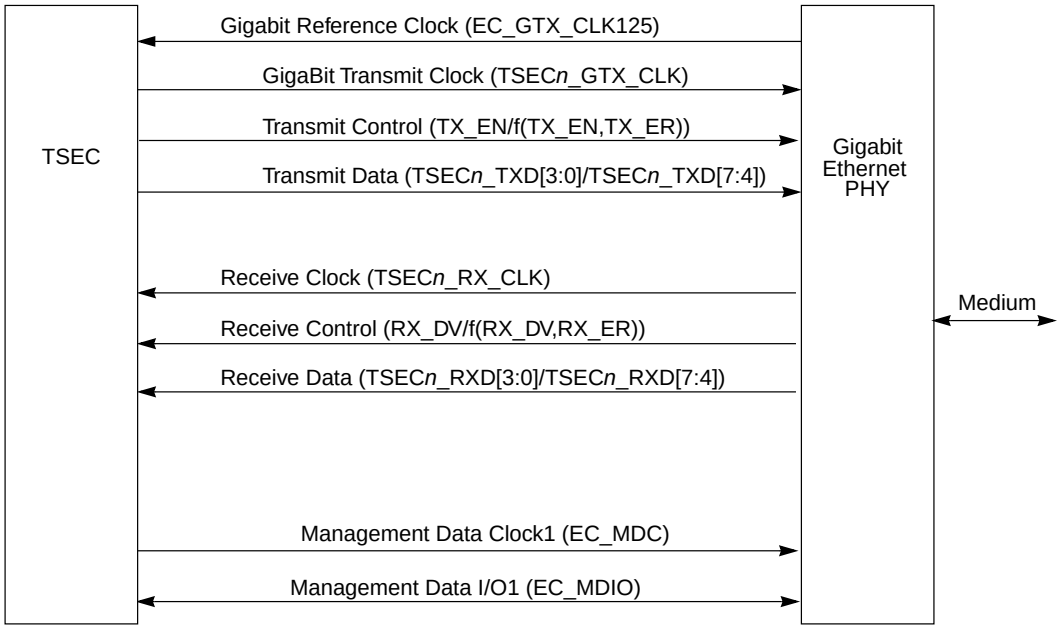
[1] The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 13-107. TSEC-GMII Connection

A GMII interface has 28 signals (TSEC*n*_GTX_CLK + EC_GTX_CLK125 included), as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY.

13.6.1.3 Reduced Gigabit Media-Independent Interface (RGMII)

This section describes the reduced gigabit media-independent interface (RGMII) intended to be used between the PHYs and the GMII MAC. The RGMII is an alternative to the IEEE802.3u MII, the IEEE802.3z GMII and the TBI. The RGMII 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 13-108 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 RGMII is implemented as defined by the RGMII specification Version 1.2a 9/22/00.

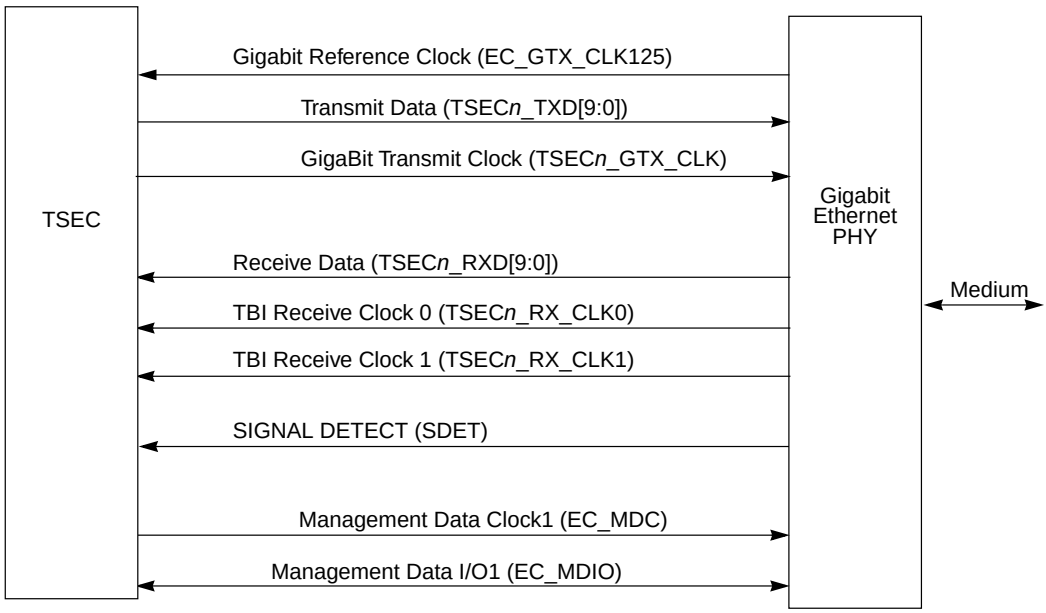


[1] The management signals (EC_MDC and EC_MDIO) are common to all of the gigabit Ethernet controllers module connections in the system, assuming that each PHY has a different management address.

Figure 13-108. TSEC-RGMII Connection

13.6.1.4 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 13-109 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.



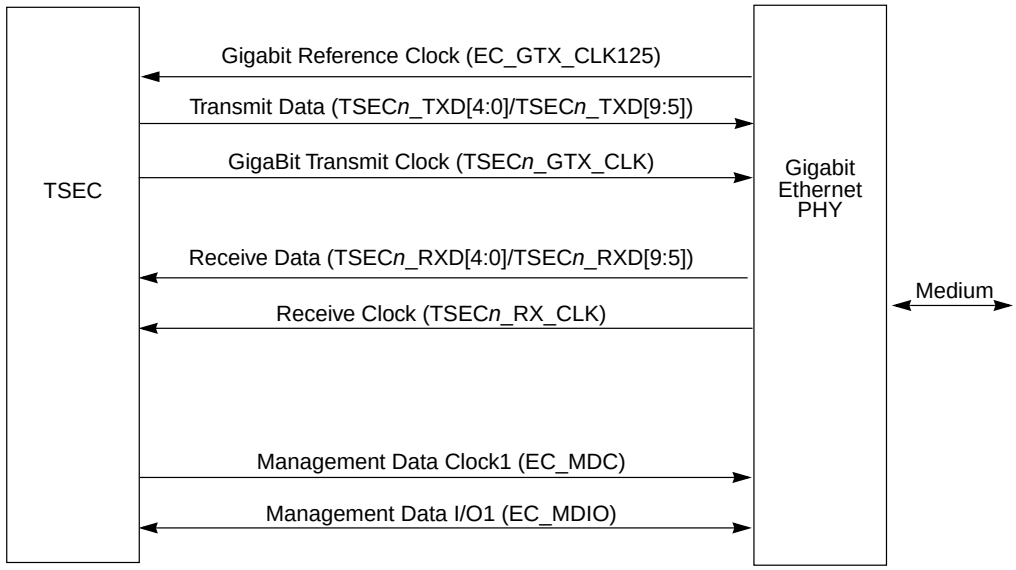
[1] The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 13-109. 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.

13.6.1.5 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 13-110 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 13-110. 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 13-106 describes the pin multiplexing for GMII, MII and TBI interfaces.

Table 13-106. 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 (TSEC*n*_)	**I/O**	**No. of Pins**	**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	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 13-106. 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 13-107 describes the pin multiplexing for RGMII and RTBI interfaces.

Table 13-107. 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 13-107. 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 (TSEC*n*_)	**I/O**	**No. of Pins**	**Signals (TSEC*n*_)**	**I/O**	**No. of Pins**	**Signals (TSEC*n*_)**	**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 13-108 describes the pins shared by all interfaces.

Table 13-108. 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	

13.6.2 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.

13.6.2.1 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.

13.6.2.1.1 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.

13.6.2.1.2 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 13-2 for the register list. Table 13-109 describes the minimum steps for register initialization.

Table 13-109. 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 13-109. 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 13.6.3.1, “Data Buffer Descriptor,” for more information.

13.6.2.2 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 gratefully 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

13.6.2.3 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 13-3. 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.

13.6.2.4 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.

13.6.2.5 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.

13.6.2.6 Frame Recognition

The Ethernet controller performs frame recognition using destination address (DA) recognition. A frame can be rejected or accepted based on the outcome.

13.6.2.6.1 Destination Address Recognition

The Ethernet controller can also perform the frame filtering using the traditional destination address (DA) recognition methods.

Figure 13-111 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.

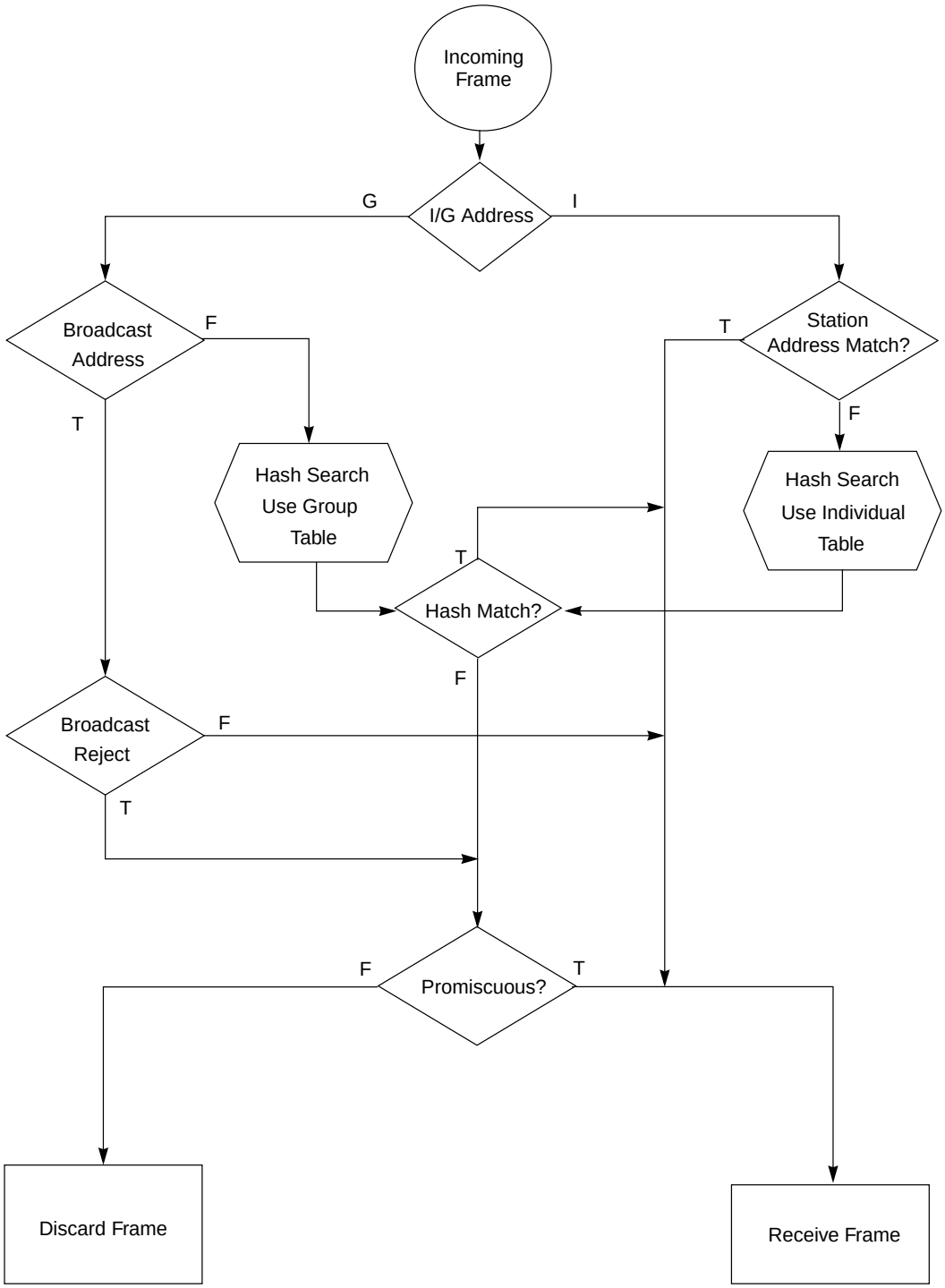


Figure 13-111. Ethernet Address Recognition Flowchart

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 13.5.3.4.1, “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.

13.6.2.6.2 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.

13.6.2.7 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 13-110 lists the flow-control frame structure.

Table 13-110. 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.

13.6.2.8 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 13-111.

- Obtain data from the RxBD 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 RxBD; thus, it is important to check more than just one RxBD during interrupt handling. Typically, all RxBDs in the interrupt handler are processed until one is found with E set. Because the TSEC pre-fetches BDs, the BD table must be big enough so that there is always another empty BD to pre-fetch. See Table 13-112.
- Clear any set halt bits in TSTAT and RSTAT registers, or DMACTRL[GTS] and DMACTRL[GRS].
- Continue normal execution.

Table 13-111. 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 13-112. 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” RxBD to memory before setting IEVENT[RXB0].
RXF	Receive frame: A frame was received and the last receive buffer descriptor (RxBD) of that frame was updated.	Programmable “write with response” to memory on the last RxBD before setting IEVENT[RXF0].

13.6.2.9 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.

13.6.2.10 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.

13.6.2.11 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 13-113.

Table 13-113. 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 13-114.

Table 13-114. 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 RxBD[OV], sets RxBD[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 (RxBD[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 RxBD[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

13.6.3 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.

13.6.3.1 Data Buffer Descriptor

Data buffers are used in the transmission and reception of Ethernet frames (see Figure 13-112). 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

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 13-113), 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.

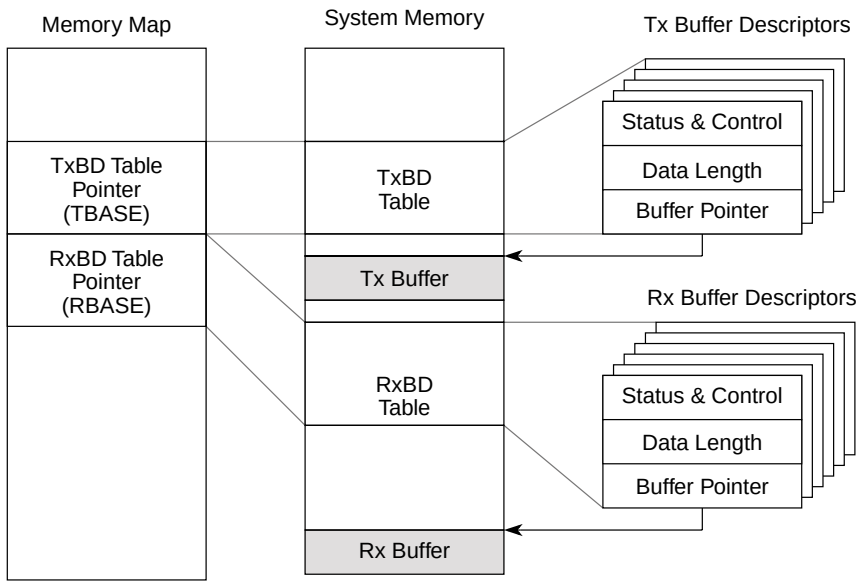


Figure 13-112. Example of TSEC Memory Structure for BD

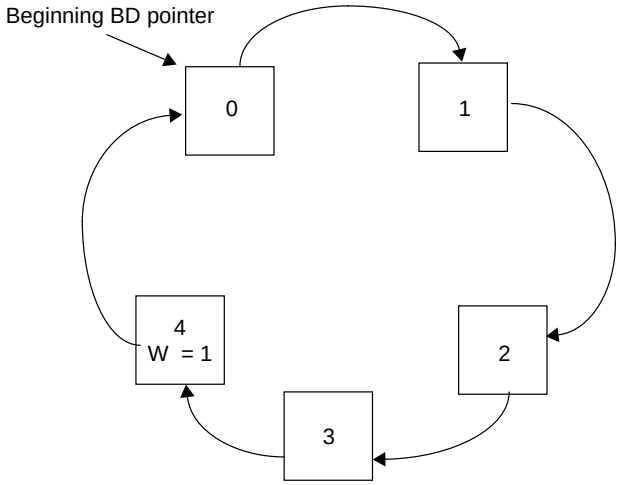


Figure 13-113. Buffer Descriptor Ring

13.6.3.2 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 13-114 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 13-114. Transmit Buffer Descriptor

The TxBD fields are detailed in Table 13-115.

Table 13-115. 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 13-115. 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 13-115. 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.

13.6.3.3 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 13-115 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 13-115. Receive Buffer Descriptor

Table 13-116 describes the fields of the RxBD.

Table 13-116. 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 13-116. 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.

13.7 Application Information

13.7.1 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.

13.7.1.1 MII Interface Mode

Table 13-117 describes the pin configurations required for MII interface mode.

Table 13-117. 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 13-118 describes the shared pins of the MII interface.

Table 13-118. Shared MII Pins

TSEC Signals	I/O	No. of pins	MII 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 13-119 describes the register initializations required to configure the TSEC in MII mode.

Table 13-119. MII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_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 13-119. 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 (0ptional) Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), read the MIIMSTAT register and verify that MIIMSTAT ---> [0000_0000_0000_0000_0000_0010_0001_0000]
Check to see if PHY has completed Auto-Negotiation Setting up the MII Mgmt for a read cycle to PHY's MII Mgmt register (write the PHY's address and Register address), MIIMADD[0000_0000_0000_0000_0000_0010_0000_0010] the Phy Status control register is at address 0x2 and lets say the PHY Address is 0x2
Perform an MII Mgmt read cycle of Status Register Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_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]

Table 13-119. 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_0000_0101]

13.7.1.2 GMII Interface Mode

Table 13-120 describes the pin configurations required for GMII interface mode.

Table 13-120. GMII Interlace 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 13-120. 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 13-121 describes the shared pins of the GMII interface.

Table 13-121. 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 13-122 describes the register initializations required to configure the TSEC in GMII mode.

Table 13-122. 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 13-122. GMII Mode Register Initialization Steps (continued)

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
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 (0ptional) Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), read the MIIMSTAT register and verify that MIIMSTAT ---> [0000_0000_0000_0000_0000_0010_0001_0000]
Check to see if PHY has completed Auto-Negotiation Setting up the MII Mgmt for a read cycle to PHY's MII Mgmt register (write the PHY's address and Register address), MIIMADD[0000_0000_0000_0000_0000_0010_0000_0010] the Phy Status control register is at address 0x2 and lets say the PHY Address is 0x2
Perform an MII Mgmt read cycle of Status Register Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_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 13-122. 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]

13.7.1.3 TBI Interface Mode

Table 13-123 describes the pin configurations required for TBI interface mode.

Table 13-123. 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 13-123. 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 13-124 describes the shared pins for the TBI interface.

Table 13-124. 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 13-125 describes the register initializations required to configure the TSEC in TBI mode.

Table 13-125. 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 13-125. 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 (0ptional) Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), read the MIIMSTAT register and verify that MIIMSTAT ---> [0000_0000_0000_0000_0000_0010_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 13-125. TBI Mode Register Initialization Steps (continued)

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]

13.7.1.4 RGMII Interface Mode

Table 13-126 shows the pins configurations required for RGMII interface mode.

Table 13-126. 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 13-126. 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 13-127 describes the shared pins for the RGMII interface.

Table 13-127. 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 13-128 describes the register initializations required to configure the TSEC in RGMII mode.

Table 13-128. RGMII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_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 13-128. RGMII Mode Register Initialization Steps (continued)

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
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 (0ptional) Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), read the MIIMSTAT register and verify that MIIMSTAT ---> [0000_0000_0000_0000_0000_0010_0001_0000]
Check to see if PHY has completed Auto-Negotiation Setting up the MII Mgmt for a read cycle to PHY's MII Mgmt register (write the PHY's address and Register address), MIIMADD[0000_0000_0000_0000_0000_0010_0000_0010] the Phy Status control register is at address 0x2 and lets say the PHY Address is 0x2
Perform an MII Mgmt read cycle of Status Register Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_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 13-128. 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]

13.7.1.5 RTBI Interface Mode

Table 13-129 describes the pin configurations required for RTBI interface mode.

Table 13-129. 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 13-129. 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 13-130 describes the shared pins for the RTBI interface.

Table 13-130. 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 13-131 describes the register initializations required to configure the TSEC in RTBI mode.

Table 13-131. 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 (0ptional) Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), read the MIIMSTAT register and verify that MIIMSTAT ---> [0000_0000_0000_0000_0000_0010_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 13-131. 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 14
DMA Controller

14.1 Introduction

The DMA controller transfers blocks of data between RapidIO and the local address space, independent of the e500 core or external hosts.

14.1.1 Block Diagram

Figure 14-1 depicts the block diagram of the DMA controller.

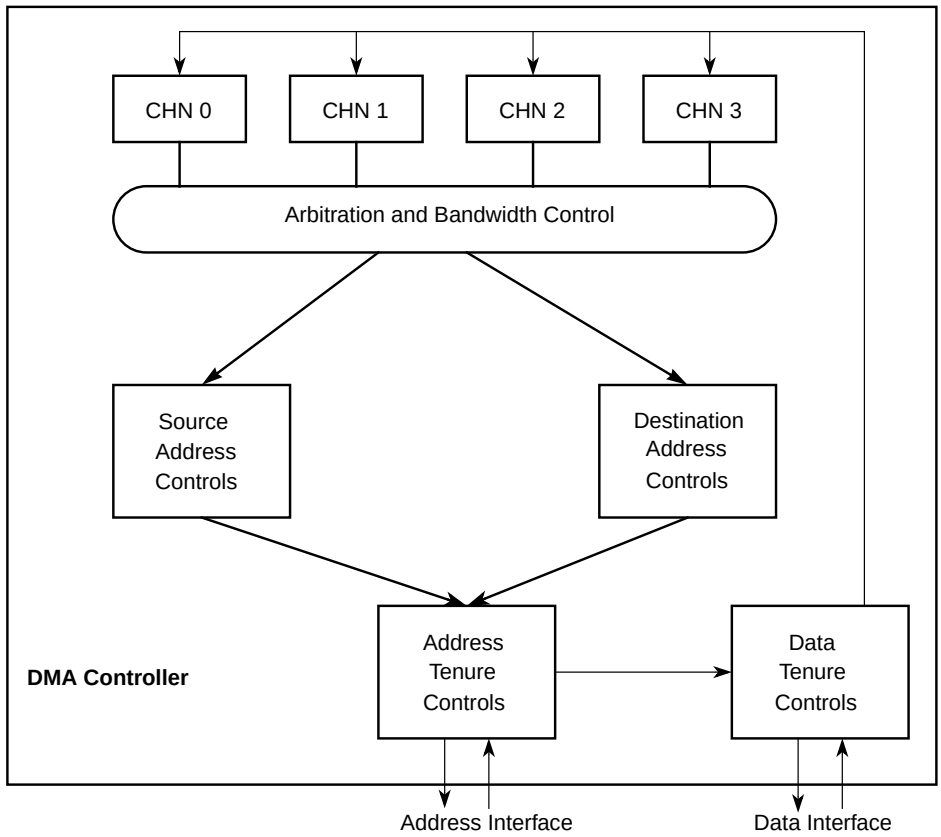


Figure 14-1. DMA Block Diagram

14.1.2 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 14-1 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.

14.1.3 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 $\overline{\text{DMA_DREQ}}$, $\overline{\text{DMA_DACK}}$, and $\overline{\text{DMA_DDONE}}$

14.1.4 Modes of Operation

The MPC8560 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 14-1 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 14-2 before starting a transfer.

- Chaining mode. Software must initialize descriptors in memory and the required fields as described in Table 14-2 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 MR*n*[CA]. The DMA controller terminates all outstanding transfers initiated by the channel without generating any errors before entering an idle state.

Table 14-1. 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 14-2 describes bit settings required for each DMA mode of operation

Table 14-2. DMA Mode Bit Settings

Modes with Features	MR*n*[XFE]	MR*n*[CTM]	MR*n*[SRW]	MR*n*[EM]	MR*n*[CDSM/SWSM]	MR*n*[EMS_EN]
Basic Direct Modes						
Basic direct	0	1	0	0	0	0
Basic direct external control	0	1	0	0	0	1

Table 14-2. DMA Mode Bit Settings (continued)

Modes with Features	MR*n*[XFE]	MR*n*[CTM]	MR*n*[SRW]	MR*n*[EM]	MR*n*[CDSM/SWSM]	MR*n*[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 14.4, “Functional Description,” for details on these modes.

Figure 14-2 depicts the general DMA operational flow chart.

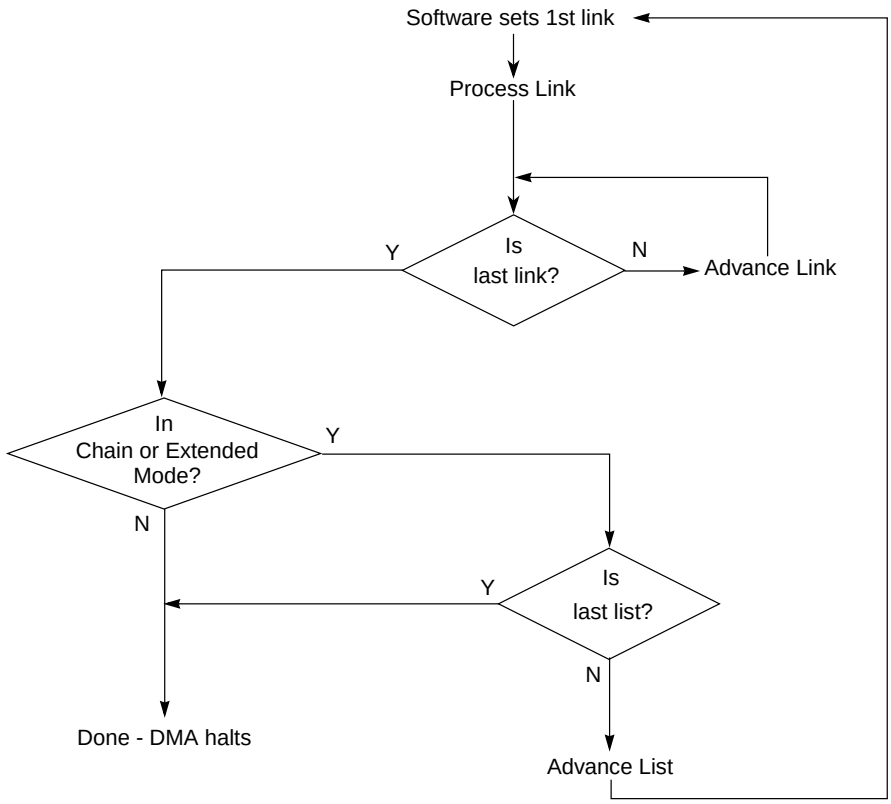


Figure 14-2. DMA Operational Flow Chart

14.2 External Signal Description

This section describes the DMA signals.

14.2.1 Signal Overview

Figure 14-3 summarizes the DMA controller signals.

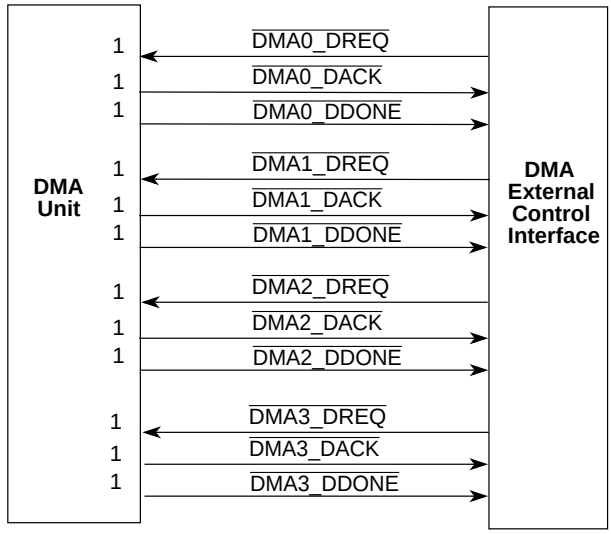


Figure 14-3. DMA Signal Summary

Note that the three DMA signals for DMA channel 3 are multiplexed with the IRQ9–11 signals on the MPC8560 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 17.3.1.10, “Alternate Function Signal Multiplex Control Register (PMUXCR).”

14.2.2 Detailed Signal Descriptions

The DMA signals are described in Table 14-3.

Table 14-3. DMA Signals—Detailed Signal Descriptions

Signal	I/O	Description	
$\overline{\text{DMA}n\text{_DREQ}}$ 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 $\overline{\text{DMAn_DREQ}}$ causes MRn[CS] to be set, thereby activating the corresponding DMA channel.	
		State Meaning	Asserted—Assertion of $\overline{\text{DMA}n\text{_DREQ}}$ while $\overline{\text{DMA}n\text{_DACK}}$ is negated causes a new transfer to start OR resumes a paused transfer if the EMP_EN bit is set. Assertion of $\overline{\text{DMA}n\text{_DREQ}}$ while $\overline{\text{DMA}n\text{_DACK}}$ is asserted results in an illegal condition Negated—Negation of $\overline{\text{DMA}n\text{_DREQ}}$ while $\overline{\text{DMA}n\text{_DACK}}$ is asserted has no effect. Negation of $\overline{\text{DMA}n\text{_DREQ}}$ before the assertion of $\overline{\text{DMA}n\text{_DACK}}$ results in an illegal condition
		Timing	Assertion—Can be asserted asynchronously. Negation— Must remain asserted at least until the assertion of the corresponding $\overline{\text{DMA}n\text{_DACK}}$.

Table 14-3. 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 timeafter 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 MR*n*[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, $\overline{\text{DMAn_DDONE}}$ is asserted after the negation of the final $\overline{\text{DMAn_DACK}}$. Negation—This signal is negated after the assertion of $\overline{\text{DMA}n\text{_DREQ}}$ for the next transfer.

14.3 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.

14.3.1 Module Memory Map

Figure 14-4 depicts the address map of DMA memory.

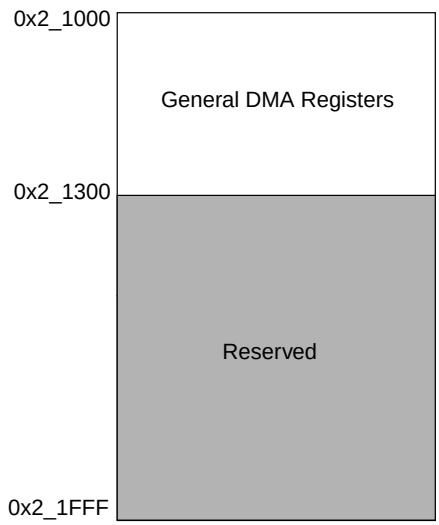


Figure 14-4. DMA Memory Map

Table 14-4 lists the classes and types of DMA registers.

Table 14-4. DMA Register Summary

Offset	Register	Access	Reset	Section/Page
General Registers				
0x2_1100	DMA 0 mode register	R/W	0x0000_0000	14.3.2.1/14-10
0x2_1104	DMA 0 status register	Special	0x0000_0000	14.3.2.2/14-13
0x2_1108	Reserved	—	—	—
0x2_110C	DMA 0 current link descriptor address register	R/W	0x0000_0000	14.3.2.3/14-14
0x2_1110	DMA 0 source attributes register	R/W	0x0000_0000	14.3.2.4/14-16
0x2_1114	DMA 0 source address register	R/W	0x0000_0000	14.3.2.5/14-18
0x2_1118	DMA 0 destination attributes register	R/W	0x0000_0000	14.3.2.6/14-19
0x2_111C	DMA 0 destination address register	R/W	0x0000_0000	14.3.2.7/14-21
0x2_1120	DMA 0 byte count register	R/W	0x0000_0000	14.3.2.8/14-23
0x2_1124	Reserved	—	—	—
0x2_1128	DMA 0 next link descriptor address register	R/W	0x0000_0000	14.3.2.9/14-23
0x2_1130	Reserved	—	—	—
0x2_1134	DMA 0 current list descriptor address register	R/W	0x0000_0000	14.3.2.10/14-24
0x2_1138	Reserved	—	—	—
0x2_113C	DMA 0 next list descriptor address register	R/W	0x0000_0000	14.3.2.11/14-25
0x2_1140	DMA 0 source stride register	R/W	0x0000_0000	14.3.2.12/14-26
0x2_1144	DMA 0 destination stride register	R/W	0x0000_0000	14.3.2.13/14-26
0x2_1180	DMA 1 mode register	R/W	0x0000_0000	14.3.2.1/14-10
0x2_1184	DMA 1 status register	Special	0x0000_0000	14.3.2.2/14-13
0x2_1188	Reserved	—	—	—
0x2_118C	DMA 1 current link descriptor address register	R/W	0x0000_0000	14.3.2.3/14-14
0x2_1190	DMA 1 source attributes register	R/W	0x0000_0000	14.3.2.4/14-16
0x2_1194	DMA 1 source address register	R/W	0x0000_0000	14.3.2.5/14-18
0x2_1198	DMA 1 destination attributes register	R/W	0x0000_0000	14.3.2.6/14-19
0x2_119C	DMA 1 destination address register	R/W	0x0000_0000	14.3.2.7/14-21
0x2_11A0	DMA 1 byte count register	R/W	0x0000_0000	14.3.2.8/14-23
0x2_11A4	Reserved	—	—	—
0x2_11A8	DMA 1 next link descriptor address register	R/W	0x0000_0000	14.3.2.9/14-23
0x2_11B0	Reserved	—	—	—
0x2_11B4	DMA 1 current list descriptor address register	R/W	0x0000_0000	14.3.2.10/14-24
0x2_11B8	Reserved	—	—	—
0x2_11BC	DMA 1 next list descriptor address register	R/W	0x0000_0000	14.3.2.11/14-25
0x2_11C0	DMA 1 source stride register	R/W	0x0000_0000	14.3.2.12/14-26

Table 14-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x2_11C4	DMA 1 destination stride register	R/W	0x0000_0000	14.3.2.13/14-26
0x2_1200	DMA 2 mode register	R/W	0x0000_0000	14.3.2.1/14-10
0x2_1204	DMA 2 status register	Special	0x0000_0000	14.3.2.2/14-13
0x2_1208	Reserved	—	—	—
0x2_120C	DMA 2 current link descriptor address register	R/W	0x0000_0000	14.3.2.3/14-14
0x2_1210	DMA 2 source attributes register	R/W	0x0000_0000	14.3.2.4/14-16
0x2_1214	DMA 2 source address register	R/W	0x0000_0000	14.3.2.5/14-18
0x2_1218	DMA 2 destination attributes register	R/W	0x0000_0000	14.3.2.6/14-19
0x2_121C	DMA 2 destination address register	R/W	0x0000_0000	14.3.2.7/14-21
0x2_1220	DMA 2 byte count register	R/W	0x0000_0000	14.3.2.8/14-23
0x2_1224	Reserved	—	—	—
0x2_1228	DMA 2 next link descriptor address register	R/W	0x0000_0000	14.3.2.9/14-23
0x2_1230	Reserved	—	—	—
0x2_1234	DMA 2 current list descriptor address register	R/W	0x0000_0000	14.3.2.10/14-24
0x2_1238	Reserved	—	—	—
0x2_123C	DMA 2 next list descriptor address register	R/W	0x0000_0000	14.3.2.11/14-25
0x2_1240	DMA 2 source stride register	R/W	0x0000_0000	14.3.2.12/14-26
0x2_1244	DMA 2 sestination stride register	R/W	0x0000_0000	14.3.2.13/14-26
0x2_1280	DMA 3 mode register	R/W	0x0000_0000	14.3.2.1/14-10
0x2_1284	DMA 3 status register	Special	0x0000_0000	14.3.2.2/14-13
0x2_1288	Reserved	—	—	—
0x2_128C	DMA 3 current link descriptor address register	R/W	0x0000_0000	14.3.2.3/14-14
0x2_1290	DMA 3 source attributes register	R/W	0x0000_0000	14.3.2.4/14-16
0x2_1294	DMA 3 source address register	R/W	0x0000_0000	14.3.2.5/14-18
0x2_1298	DMA 3 destination attributes register	R/W	0x0000_0000	14.3.2.6/14-19
0x2_129C	DMA 3 destination address register	R/W	0x0000_0000	14.3.2.7/14-21
0x2_12A0	DMA 3 byte count register	R/W	0x0000_0000	14.3.2.8/14-23
0x2_12A4	Reserved	—	—	—
0x2_12A8	DMA 3 next link descriptor address register	R/W	0x0000_0000	14.3.2.9/14-23
0x2_12B0	Reserved	—	—	—
0x2_12B4	DMA 3 current list descriptor address register	R/W	0x0000_0000	14.3.2.10/14-24
0x2_12B8	Reserved	—	—	—
0x2_12BC	DMA 3 next list descriptor address register	R/W	0x0000_0000	14.3.2.11/14-25
0x2_12C0	DMA 3 source stride register	R/W	0x0000_0000	14.3.2.12/14-26

Table 14-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x2_12C4	DMA 3 destination stride register	R/W	0x0000_0000	14.3.2.13/14-26
0x2_1300	DMA general status register	Read	0x0000_0000	14.3.2.14/14-27

14.3.2 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 14-5 that describes the register.

14.3.2.1 Mode Registers (MR*n*)

The mode register allows software to start a DMA transfer and to control various DMA transfer characteristics.

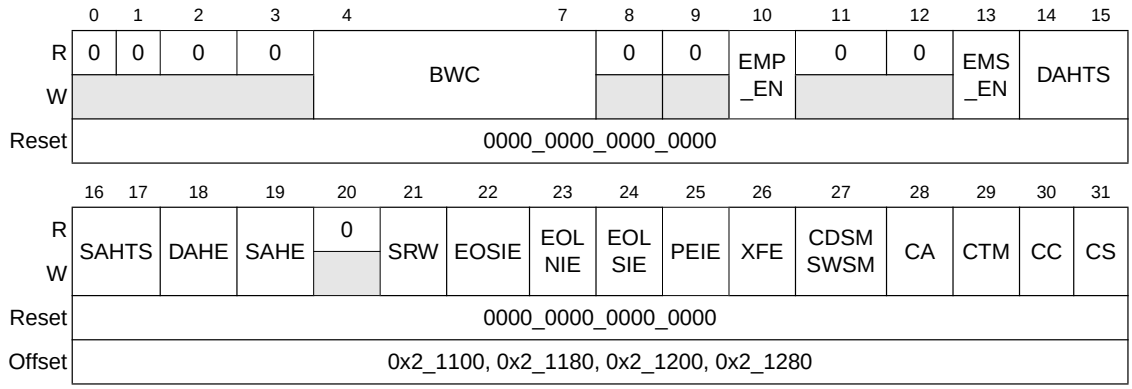


Figure 14-5. DMA Mode Registers (MR*n*)

Table 14-5 describes the bits of the mode registers.

Table 14-5. 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 Diasable 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 MRn[SAHTS]. Hardware only supports aligned transfers for this feature.

Table 14-5. 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 14-5. 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.

14.3.2.2 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 14-6. Status Registers (SR*n*)

Table 14-6 describes the bits of the status registers.

Table 14-6. 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 14-6. 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 CLNDARn[EOSIE] is set, this bit gets set and an interrupt is generated. In direct mode, if MRn[EOSIE] is set, this bit gets set and an interrupt is generated. (Bit reset, write 1 to clear)
31	EOLSI	End-of-list interrupt. After transferring the last block of data in the last list descriptor, if MR*n*[EOLSIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)

14.3.2.3 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 NLNDARn[EOLND] in the next link descriptor address register is examined. If NLNDARn[EOLND] is clear, the DMA controller reads in the new current link descriptor for processing. If NLNDARn[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 NLSDARn[EOLSD] in the next list descriptor address register. If NLSDARn[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 NLSDARn[EOLSD] is set all DMA transfers are complete and the DMA controller halts. Figure 14-7 depicts the process of Basic Chaining Mode DMA operation.

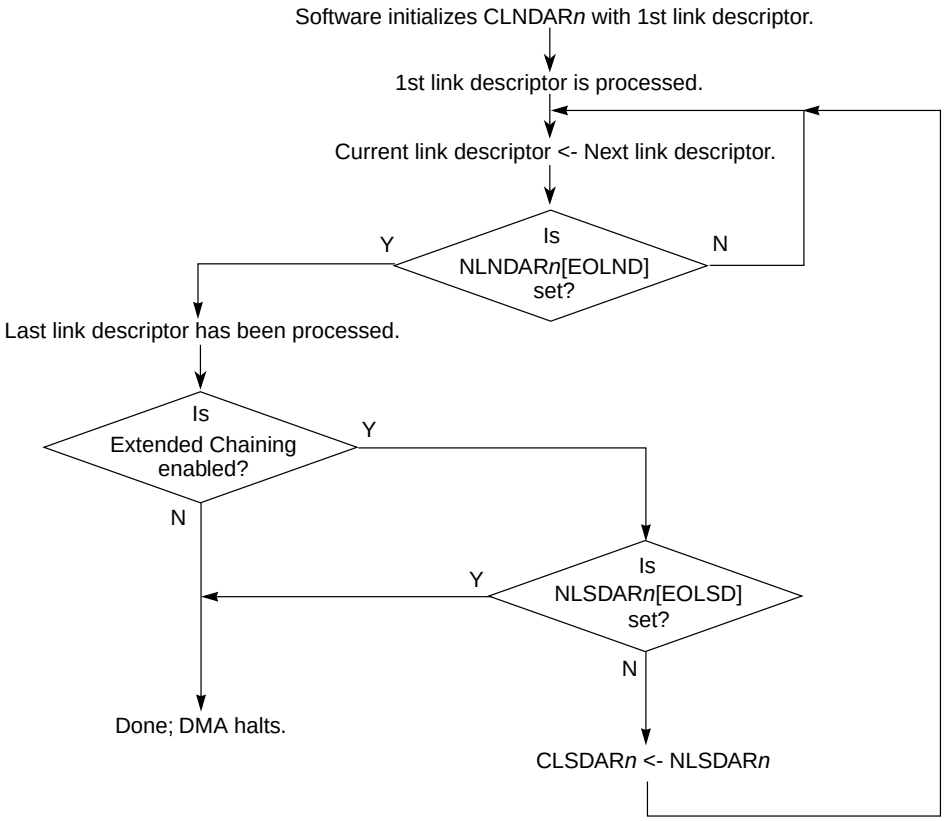


Figure 14-7. Basic Chaining Mode Flow Chart

Figure 14-8 describes the bits of the current link descriptor address registers.

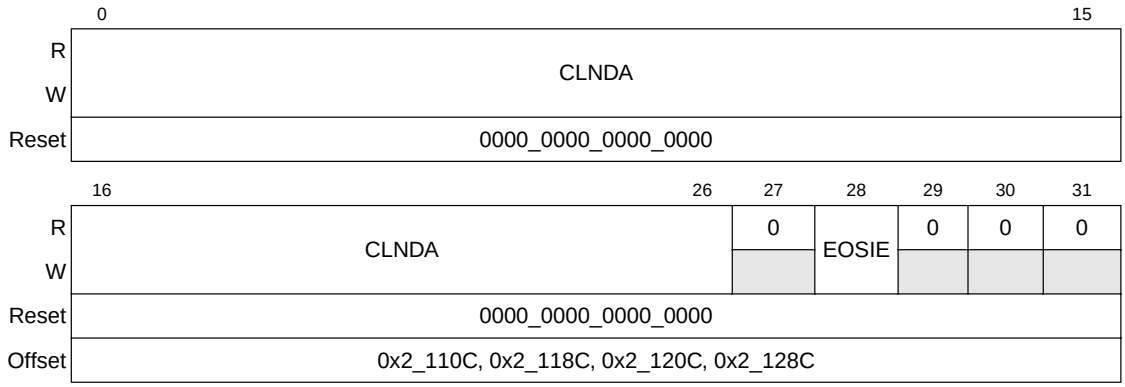


Figure 14-8. Current Link Descriptor Address Registers (CLNDAR*n*)

Table 14-7 describes the bits of the current link descriptor address registers.

Table 14-7. 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

14.3.2.4 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.

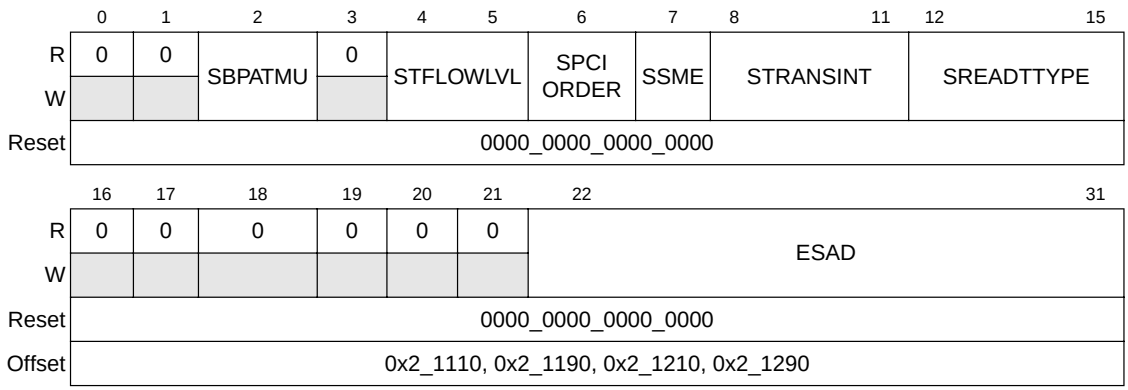


Figure 14-9. Source Attributes Registers (SATR*n*)

Table 14-8. 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 14-8. 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 14-8. 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.

14.3.2.5 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.

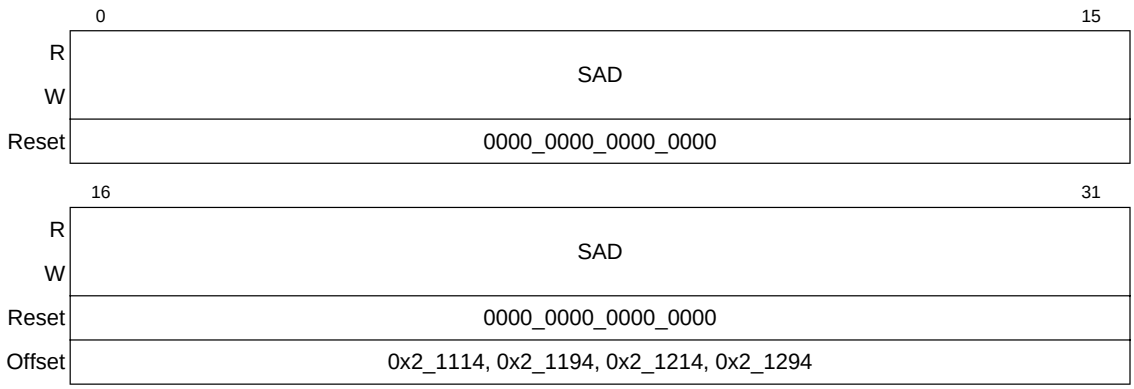


Figure 14-10. Source Address Registers (SAR*n*)

Table 14-9. 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.

14.3.2.5.1 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.

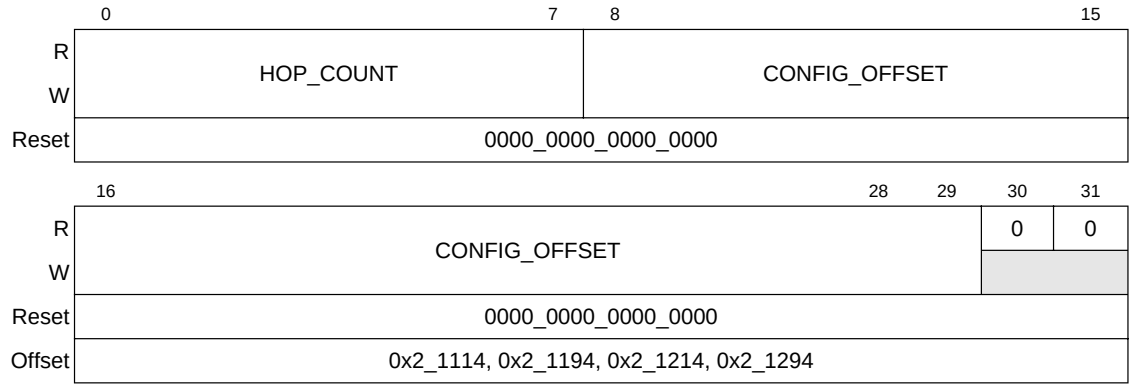


Figure 14-11. Source Address Registers for RapidIO Maintenance Reads (SARM*n*)

Table 14-10. 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

14.3.2.6 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.

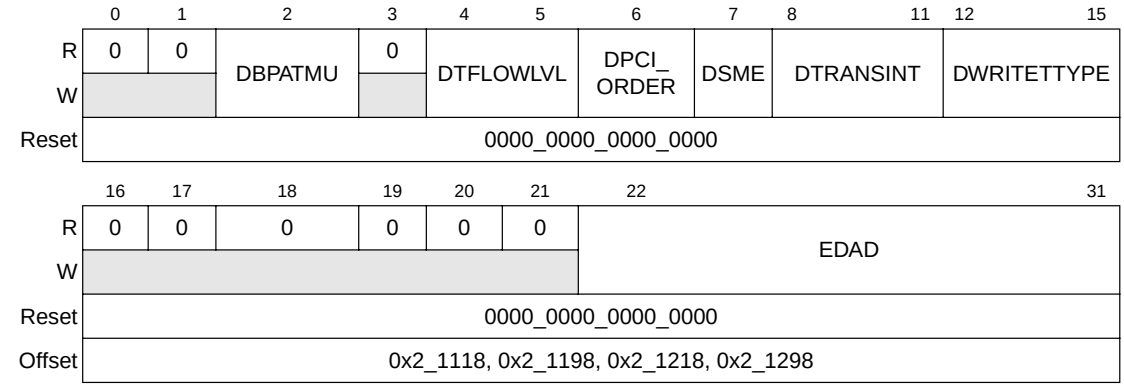


Figure 14-12. Destination Attributes Registers (DATR*n*)

Table 14-11. 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*[DWRITETTYPE] 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 14-11. DATR*n* Field Descriptions (continued)

Bits	Name	Description
12–15	DWRITETTYPE	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.

14.3.2.7 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.

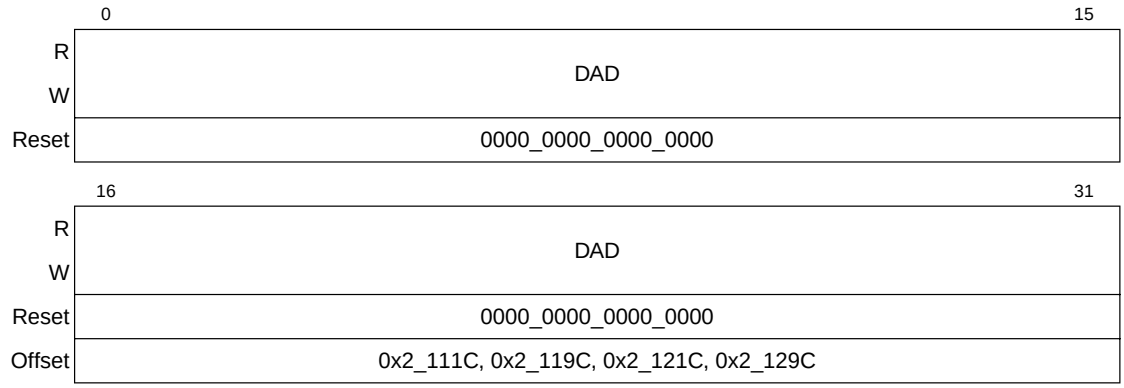


Figure 14-13. Destination Address Registers (DAR*n*)

Table 14-12. 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.

14.3.2.7.1 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.

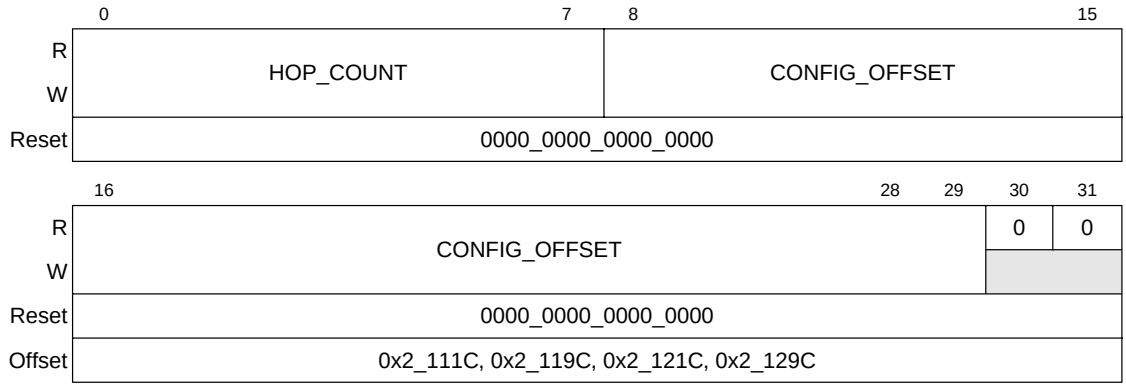


Figure 14-14. Destination Address Registers For RapidIO Maintenance Writes (DARM*n*)

Table 14-13. 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

14.3.2.8 Byte Count Registers (BCR*n*)

The byte count register contains the number of bytes to transfer.

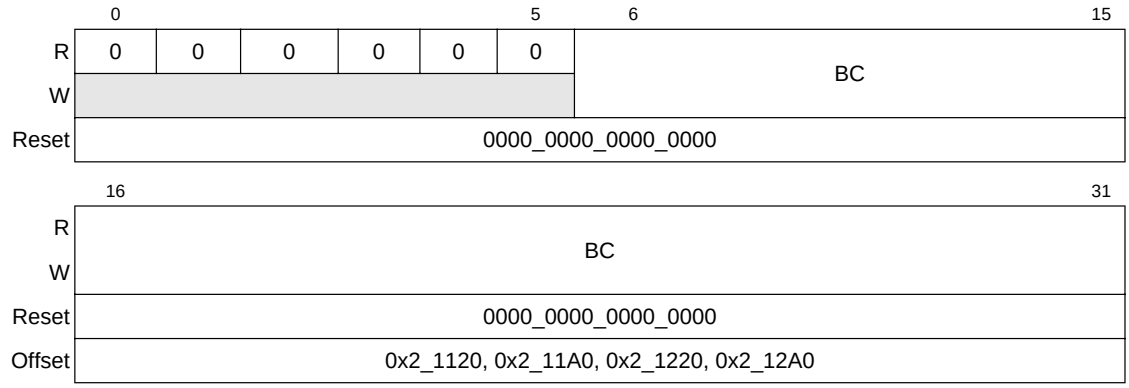


Figure 14-15. Byte Count Registers (BCR*n*)

Table 14-14. 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.

14.3.2.9 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.

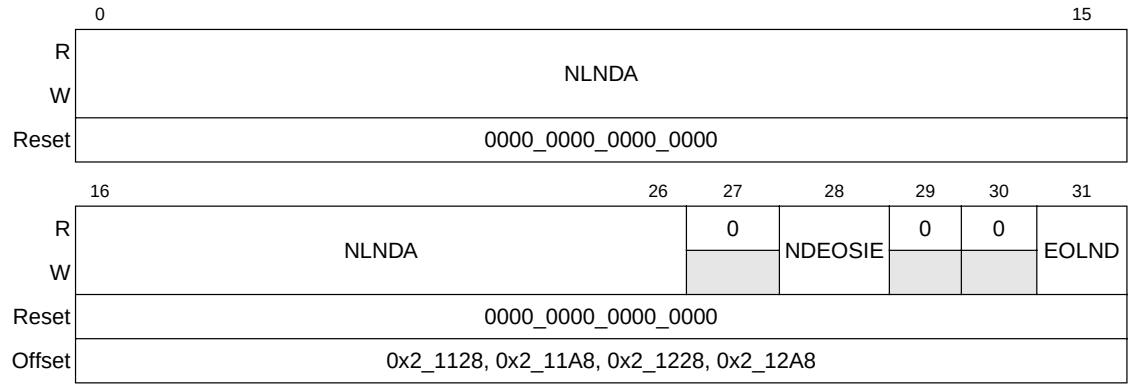


Figure 14-16. Next Link Descriptor Address Registers (NLNDAR*n*)

Table 14-16. 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.

14.3.2.10 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.

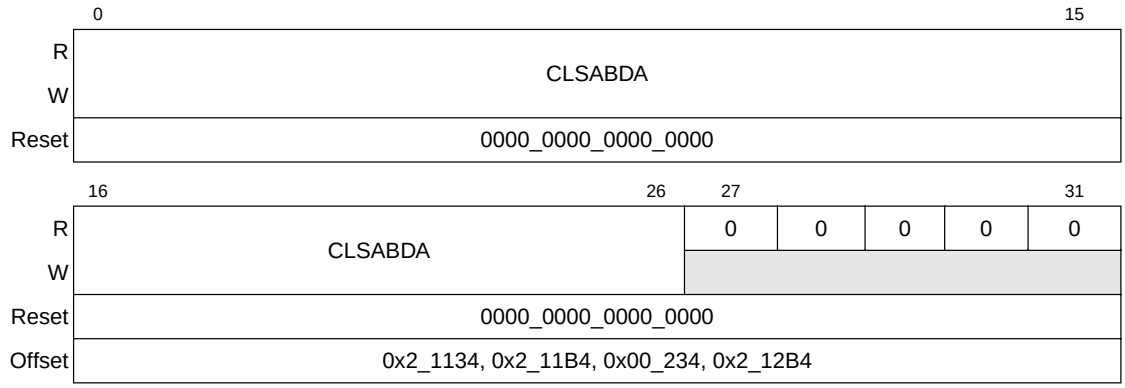


Figure 14-17. Current list Descriptor Address Registers (CLSDAR*n*)

Table 14-17. 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

14.3.2.11 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.

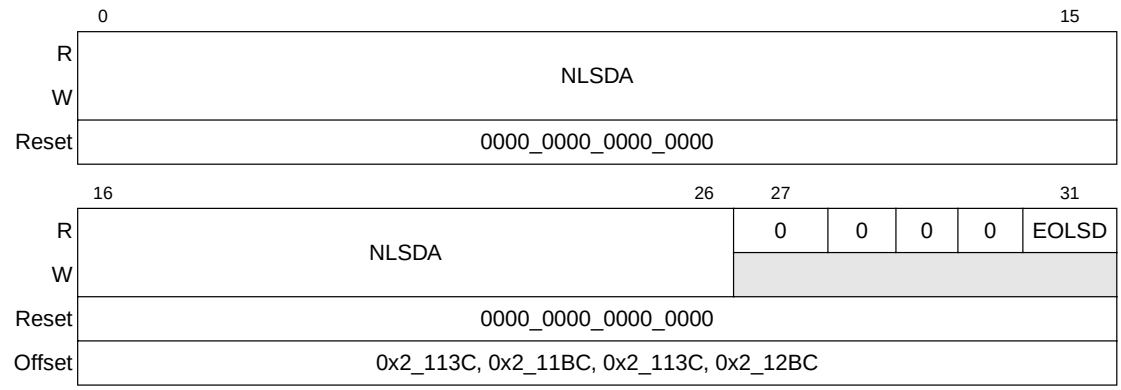


Figure 14-18. Next List Descriptor Address Registers (NLSDAR*n*)

Table 14-18. 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.

14.3.2.12 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.

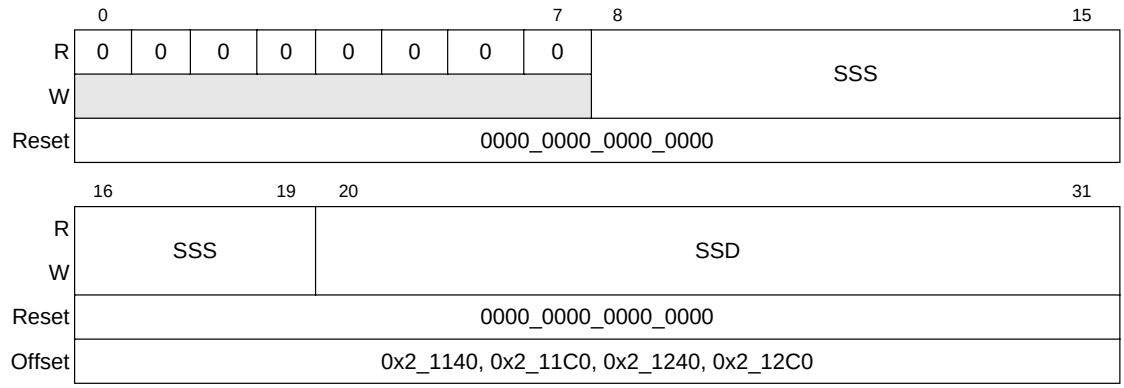


Figure 14-19. Source Stride Registers (SSR*n*)

Table 14-19. 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.

14.3.2.13 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.

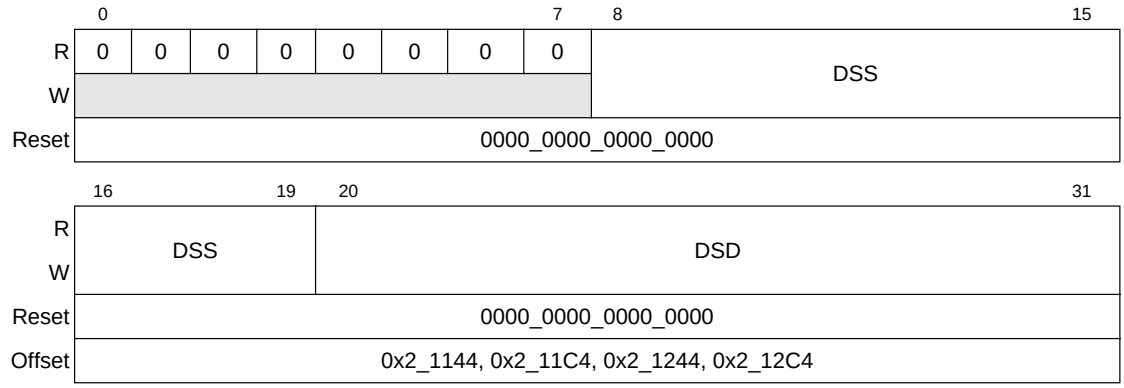


Figure 14-20. Destination Stride Registers (DSR*n*)

Table 14-20. DSR*n* Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–19	DSS	Destination stride size. Number of bytes to transfer before jumping to the next address as specified in the destination stride distance field.
20–31	DSD	Destination stride distance. The destination stride distance in bytes from start byte to start byte.

14.3.2.14 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.

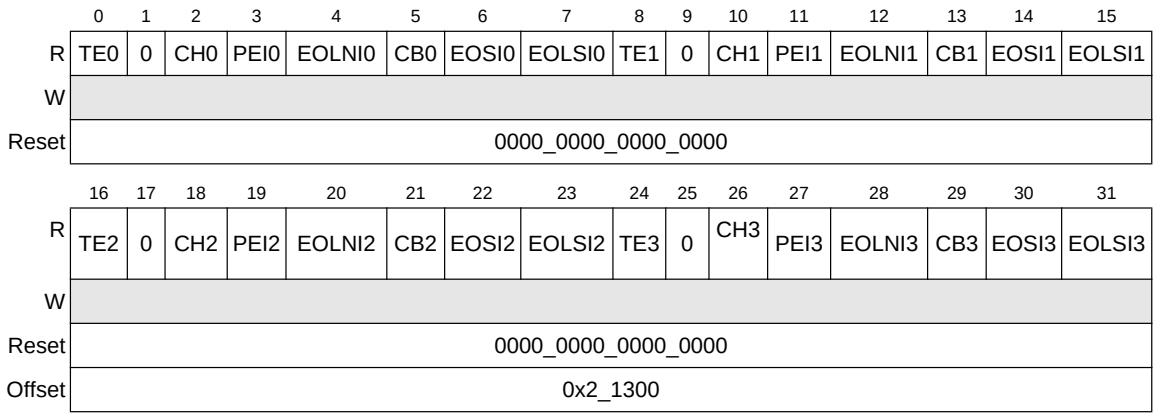


Figure 14-21. DMA General Status Register (DGSR)

Table 14-21. 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 14-21. 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

14.4 Functional Description

14.4.1 DMA Channel Operation

All DMA channels support two different modes of operation; a basic mode (MR*n*[XFE] is cleared) and an extended mode (MR*n*[XFE] is set). In both modes, a channel can be activated by clearing and setting MR*n*[CS], or via the single-write start mode using MR*n*[CDSM/SWSM] and MR*n*[SRW], or via an external control mode using MR*n*[ECS_EN]. In basic mode, the channel can be programmed in basic direct mode, or basic chaining mode. In extended mode, the channel can be programmed in extended direct mode, or extended chaining mode. Extended mode provides more capabilities 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 bandwith 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(MR*n*), 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.

14.4.1.1 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.

14.4.1.1.1 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.

14.4.1.1.2 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.

14.4.1.1.3 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, SR*n*[CB], to confirm that the specific DMA channel is idle.(SR*n*[CB] is cleared)
3. Initialize the current link descriptor address register(CLNDAR*n*) to point to the first link descriptor in memory.
4. Clear the mode register channel transfer mode bit, MR*n*[CTM], as well as MR*n*[XFE], to indicate basic chaining mode. Other control parameters may also be initialized in the mode register.
5. Clear, then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
6. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
7. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

14.4.1.1.4 Basic Chaining Single-Write Start Mode

Basic chaining single-write start mode allows a chain to be started by writing the current link descriptor address register(CLNDAR*n*). Setting MR*n*[CDSM/SWSM] in the mode register causes MR*n*[CS] to be automatically set when the current link descriptor address register is written. The sequence of events to start and complete a chain using single-write start mode is as follows:

1. Set the mode register current descriptor start mode bit, MR*n*[CDSM/SWSM], and the extended features enable bit MR*n*[XFE]. Also, clear the channel transfer mode bit, MR*n*[CTM]. This initialization indicates basic chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build link descriptor segments in memory.
3. Poll the status register channel busy bit, SR*n*[CB], to confirm that the specific DMA channel is idle.(SR*n*[CB] is cleared)
4. Initialize the current link descriptor address register(CLNDAR*n*) to point to the first descriptor segment in memory. This write automatically causes the DMA controller to begin the link descriptor fetch and set MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

14.4.1.2 Extended DMA Mode Transfer

The extended DMA mode also operates in chaining and direct mode. It offers additional capability over the basic mode by supporting striding and a more flexible descriptor structure. This additional functionality also requires a new and more complex programming model. The extended DMA mode is activated by setting MR*n*[XFE].

14.4.1.2.1 Extended Direct Mode

Extended direct mode has the same functionality as basic direct mode with the addition of stride capabilities. The bit settings are the same as in direct mode with the exception of the MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in the source stride register(SSR*n*). Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in the destination stride register(DSR*n*).

14.4.1.2.2 Extended Direct Single-Write Start Mode

Extended direct single-write start mode has the same functionality as the basic direct single-write start mode with the addition of stride capabilities. The bit settings are also the same with the exception of MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in the source stride register(SSR*n*). Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in the destination stride register(DSR*n*).

14.4.1.2.3 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(CLSDAR*n*) must be initialized to point to the first list descriptor in memory. The DMA controller loads list descriptors and link descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded. Once the current link descriptor is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer. If the current link descriptor is the last in the list, the DMA controller reads the next list descriptor in memory. The transfer is finished if the current link descriptor is the last one in the last list in memory or if an error condition occurs. The sequence of events to start and complete a transfer in extended chaining mode is as follows:

1. Build link and list descriptor segments in memory.
2. Poll the status register channel busy bit, SR*n*[CB], to confirm that the specific DMA channel is idle.(SR*n*[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.

14.4.1.2.4 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.

14.4.1.3 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 MR*n*[EMP_EN]. MR*n*[BWC] specifies how much data to allow a specific channel to transfer before entering a paused state by clearing MR*n*[CS]. 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:

- $\overline{\text{DMA_DREQ}}$, Indicates a DMA transfer start or restart from a pause request. Sets MR*n*[CS].
- $\overline{\text{DMA_DACK}}$, Indicates a DMA transfer currently in progress. SR*n*[CB] is set.
- $\overline{\text{DMA_DDONE}}$, Indicates the completion of the DMA transfer. SR*n*[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 14-22.

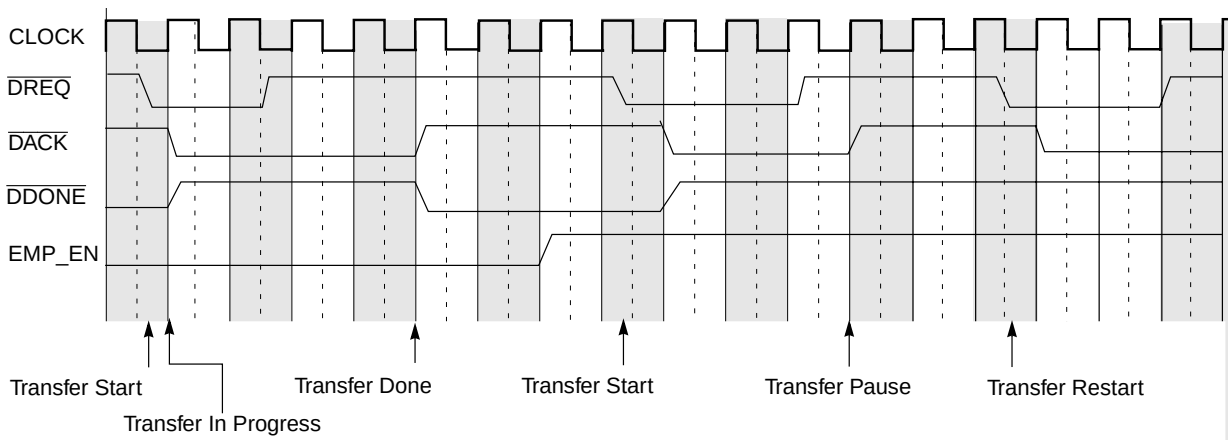


Figure 14-22. External Control Interface Timing

14.4.1.4 Channel Continue Mode for Cascading Transfer Chains

The channel continue mode(enabled when MR*n*[CC] is set) offers software the flexibility of having the DMA controller get started on descriptors that have already been programmed while software continues to build more descriptors in memory. Software can set the end-of-links descriptor(EOLND) in basic mode, or end-of-lists descriptor(EOLSD) in extended mode, to cause the channel to go into a stopped state while software continues to build other descriptors in memory. Software can then set MR*n*[CC] to force hardware to continue where it left off. channel continue is only meaningful for chaining modes, not direct mode.

If MR*n*[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 MR*n*[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.

14.4.1.4.1 Basic Mode

On a channel continue, the descriptor at the current link descriptor address register (CLNDAR*n*) is refetched to get the next link descriptor address field as updated by software. The channel halts if NLNDAR*n*[EOLND] is still set. If NLNDAR*n*[EOLND] is cleared, the next link descriptor address is copied into the CLNDAR*n* 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.

14.4.1.4.2 Extended Mode

On a channel continue, the descriptor at the current list descriptor (CLSDAR*n*) address register is refetched to get the next list descriptor address field as updated by software. The channel halts if NLSDAR*n*[EOLSD] is still set. If not, the next list descriptor address is copied into the CLSDAR*n* 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.

14.4.1.5 Channel Abort

Software can abort a previously initiated transfer by setting MR*n*[CA]. Once the DMA channel controller detects a 0 to one transition of MR*n*[CA], it finishes the current sub-block transfer and halts all further activity. The controller then waits for all previously initiated transfers from the specified channel to drain and clears SR*n*[CB]. Successful completion of a software initiated abort request can be recognized by MR*n*[CA] being set and SR*n*[CB] being cleared. Obviously, if the controller was already halted because of an error condition (SR*n*[TE] is set), or the channel has completed all transfers, then SR*n*[CB] being cleared may not signify that the controller entered a halt state due to the abort request.

14.4.1.6 Bandwidth Control

MR*n*[BWC] specifies how much data to allow a specific channel to transfer before allowing the next channel to use the shared data transfer hardware. This promotes equitable bandwidth allocation between channels. However, if only one channel is busy, hardware overrides the specified bandwidth control size value. The DMA controller allows a channel to transfer up to 1 kB at a time when no other channel is active.

14.4.1.7 Channel State

Table 14-22 defines the state of a channel based on the values of the channel start (MR*n*[CS]), channel busy (SR*n*[CB]), transfer error (SR*n*[TE]), and channel continue (MR*n*[CC]) bits.

Table 14-22. Channel State Table

MR*n*[CS]	SR*n*[CB]	SR*n*[TE]	MR*n*[CC]	Channel State
0	0	0	0	Idle state. This is the state of the bits out of reset.
0	0	0	1	Channel continue unexpected. Channel remains idle.
0	0	1	0	Error occurred after software halted the channel.
0	0	1	1	Channel Continue unexpected. Channel remains in error halt state.
0	1	0	0	Software halted channel. The channel was busy and software cleared MR*n*[CS].
0	1	0	1	Channel remains in halt state.
-	1	1	-	The Channel has encountered an error condition and it is trying to halt.
1	0	0	0	Ready to start a transfer, or transfer completed.
1	0	0	1	Continue transfer (only meaningful in chaining mode, not direct mode). In Direct mode, the Channel Continue has no effect.
1	0	1	0	Error occurred during transfer.
1	0	1	1	Channel remains in error halt state.
1	1	0	0	Transfer in progress.
1	1	0	1	Continue after reaching the end of list/link, or the first descriptor fetch after channel continue.

14.4.1.8 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 14-23 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.

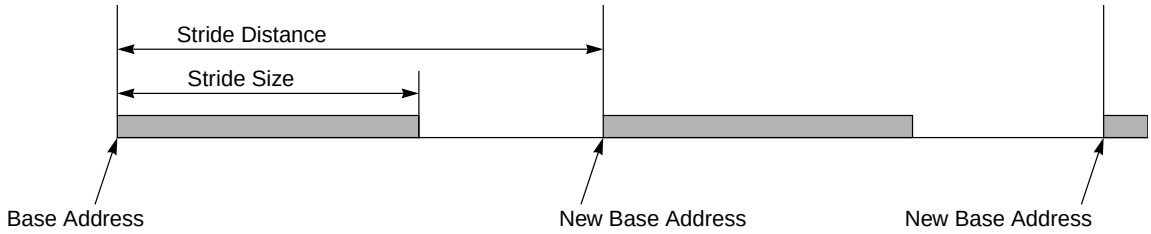


Figure 14-23. Stride Size and Stride Distance

14.4.2 DMA Transfer Interfaces

The DMA can be used to achieve data transfers across the entire memory map.

14.4.3 DMA Errors

On a transfer error, the DMA halts by setting SR*n*[TE]. On a programming error, the DMA sets SR*n*[PE] and generates an interrupt if enabled(SR*n*[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.

14.4.4 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 14-26. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor.

Table 14-23. 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 14-24. 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 14-24. 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 14-25. Link Descriptor Format

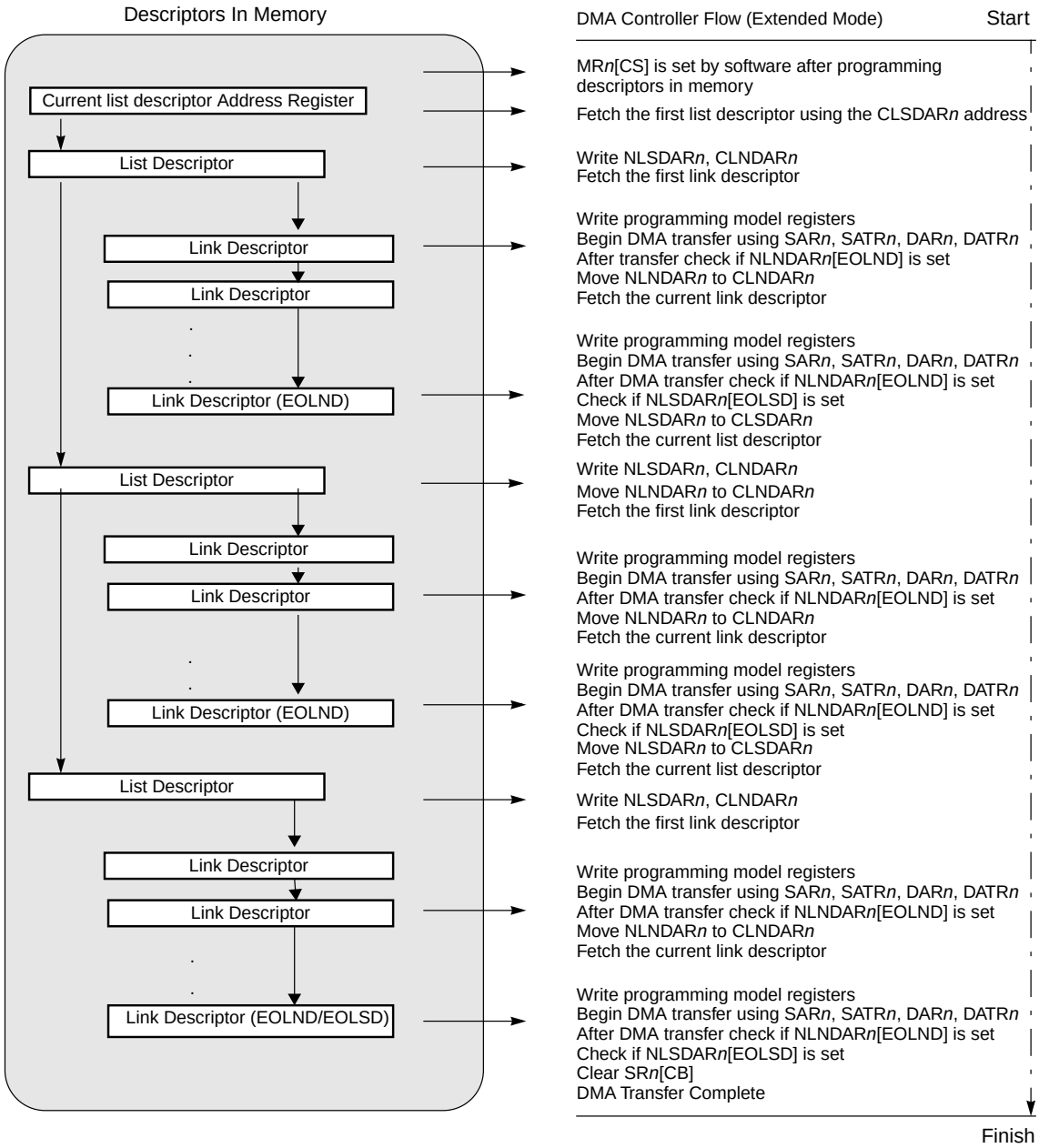


Figure 14-26. DMA Transaction Flow with DMA Descriptors

14.4.5 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 MR*n*[SAHE] is set, the source interface transfer size capability must be greater than or equal to MR*n*[SAHTS]. The source address must be aligned to the size specified by MR*n*[SAHTS].
- If MR*n*[DAHE] is set, the destination interface transfer size capability must be greater than or equal to MR*n*[DAHTS]. The destination address must be aligned to the size specified by MR*n*[DAHTS].
- The stride mode is not supported if MR*n*[DAHE] or MR*n*[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.

14.5 DMA System Considerations

Figure 14-27 depicts the most important data paths within the MPC8560. 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 14-27. Block Diagram Showing Major Data Paths

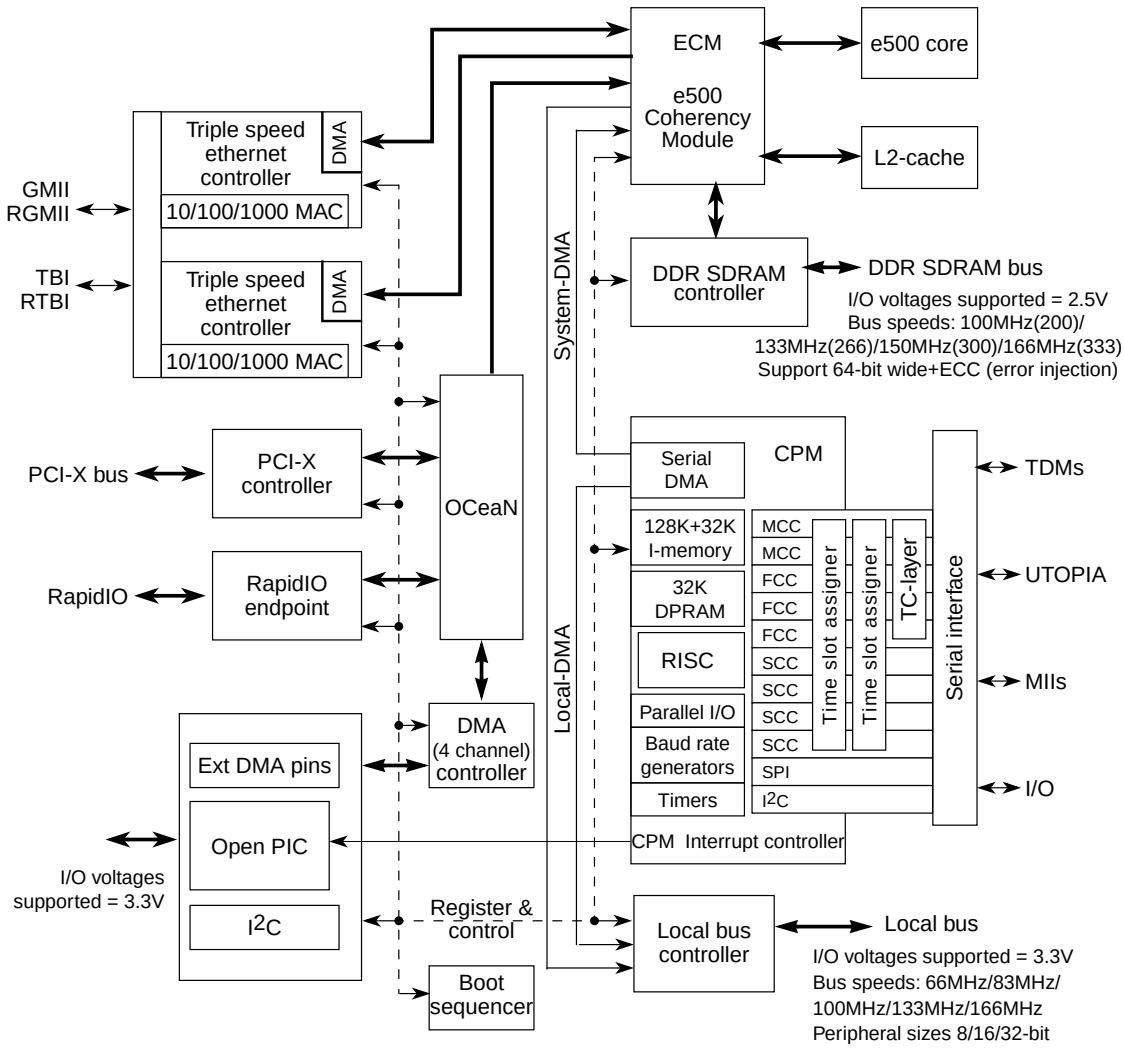


Table 14-25 lists all the DMA controllers (both captive and general-purpose) on the device and the most likely DMA targets on and off-chip. The bus controllers themselves cannot initiate DMA transfers; rather, they operate transparently with respect to both on- and off-chip DMA controllers. The codes in the table cells have the following meaning:

- Y - Supported
- NR - Possible, but not recommended. Inefficient use of system resources.
- NS - Possible, but not supported. Resulting system behavior is not defined.

14.5.1 Unusual DMA Scenarios

Table 14-25. DMA Paths

DMA Controllers		On-Chip Targets			Off-Chip Targets				
		Memory Mapped L2	CPM DPRAM	Config [1] Registers	DDR	Local Bus	PCI-X	RIO	Ethernet Buffers
On-Chip	CPM [2]	Y	NR	NS	Y	Y	Y	Y	Y
	Ethernet[2]	Y	NR	NS	Y	Y	Y	Y	Y
	4Channel[3]	Y	NR	NS	Y	Y	Y	Y	Y
Off-Chip	PCI-X Controller	Y	NR	NS	Y	Y	–	Y	Y
	RapidIO Controller	Y	NR	NS	Y	Y	Y	–	Y

1 Includes I^2C data register.

2 Captive resource. Not available to external masters.

3 Can serve external masters.

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.

14.5.1.1 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.

14.5.1.2 DMA to CPM

The CPM's dual port RAM (DPRAM) can serve as both a source and destination for general-purpose DMA transfers. However, because the CPM has its own dedicated DMA controller, using an external DMA controller to access the DPRAM makes little sense except in special circumstances (such as, possibly, during system initialization).

14.5.1.3 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 recieve 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.

14.5.1.4 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 I^2C 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.

14.5.1.5 DMA to I^2C

The I^2C controller is not transparent to DMA transfers. Observe the caveats listed in Section 14.5.1.4, “DMA to Configuration and Control Registers,” when accessing any I^2C register, including the data register (I2CDR).

Chapter 15
PCI Bus Interface

15.1 Introduction

The MPC8560 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 MPC8560 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 a a bridge between the PCI/X interface and the OCeaN switch fabric. Figure 15-1 is a high-level block diagram of the PCI/X controller.

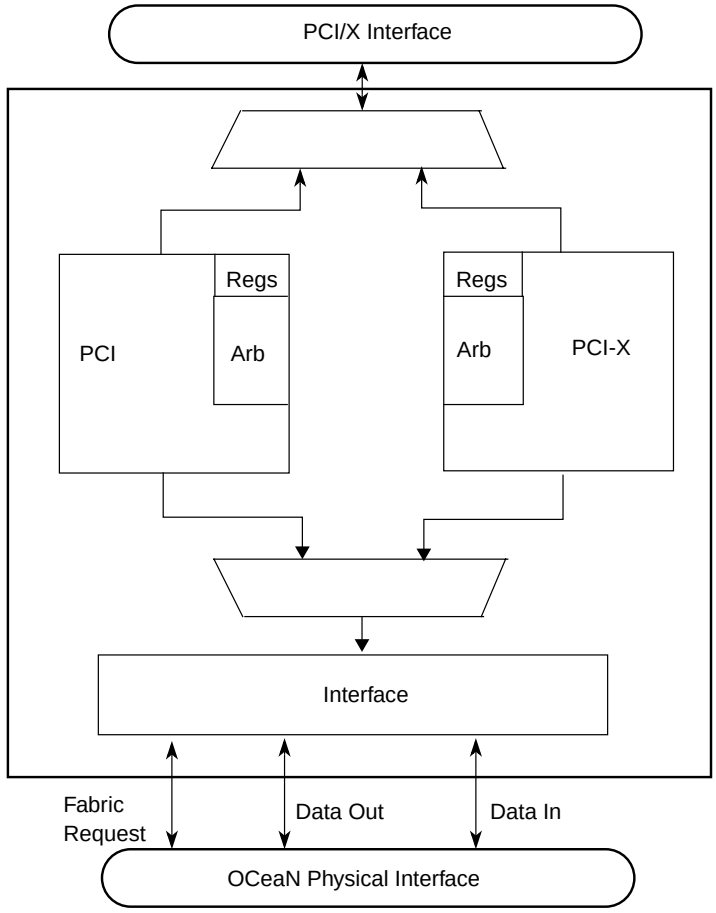


Figure 15-1. PCI/X Controller Block Diagram

15.1.1 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 MPC8560 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 MPC8560 to handle separate PCI transactions simultaneously. For example, if the MPC8560, 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 MPC8560, as a target, can accept the burst-read transfer. When the MPC8560 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 MPC8560 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 MPC8560 also supports generating PCI special-cycle and interrupt-acknowledge transactions. As a target, the MPC8560 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 MPC8560 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 15.1.3.1, “Host/Agent/Agent Lock Mode,” for more information.

In PCI agent mode, all of the PCI configuration registers in the MPC8560 can be programmed from the PCI bus. See Section 15.3.2.2, “PCI Agent Mode Configuration Register Access,” for more information.

The PCI interface provides bus arbitration for the MPC8560 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 MPC8560 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.

15.1.1.1 MPC8560 as a PCI Initiator

Upon detecting an OCeaN-to-PCI transaction, the MPC8560 requests the use of the PCI bus. For OCeaN-to-PCI bus write operations, the MPC8560 requests mastership of the PCI bus when the source completes the write operation to the OCeaN. For OCeaN-to-PCI read operations, the MPC8560 requests mastership of the PCI bus when it decodes that the access is for PCI address space.

Once granted, the MPC8560 drives the PCI address (PCI_AD[63:0]) and the bus command (PCI_C/$\overline{\text{BE}}$[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.

15.1.1.2 MPC8560 as a PCI Target

As a target, on detection of a PCI address phase, the MPC8560 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 MPC8560 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 MPC8560 supports data streaming to and from local memory. This means that the MPC8560 can accept or provide data to or from local memory as long as the internal buffers are not filled.

15.1.2 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

15.1.3 Modes of Operation

15.1.3.1 Host/Agent/Agent Lock Mode

The MPC8560 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 MPC8560for PCI host or PCI agent mode as described in Chapter 4, “Reset, Clocking, and Initialization.”

Note that in PCI agent mode, the MPC8560 ignores all PCI memory accesses (except to the memory mapped registers) until inbound address translation is enabled. See Section 15.5.1, “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 15.5.1.1, “Host Mode,” for more information.
- Agent mode.When the device powers up in agent mode, it acknowledges inbound configuration accesses. See Section 15.5.1.2, “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 15.5.1.3, “Agent Config Lock Mode,” for more information.

15.1.3.1.1 PCI/PCI-X Signal Output Hold Timing

To meet minimum output hold specifications relative to SYSCLK for both PCI and PCI-X systems, the MPC8560 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 *MPC8560 Hardware Specifications*, for more information on these values and signal timing.

15.2 External Signal Descriptions

Figure 15-2 shows the external PCI/PCI-X signals.

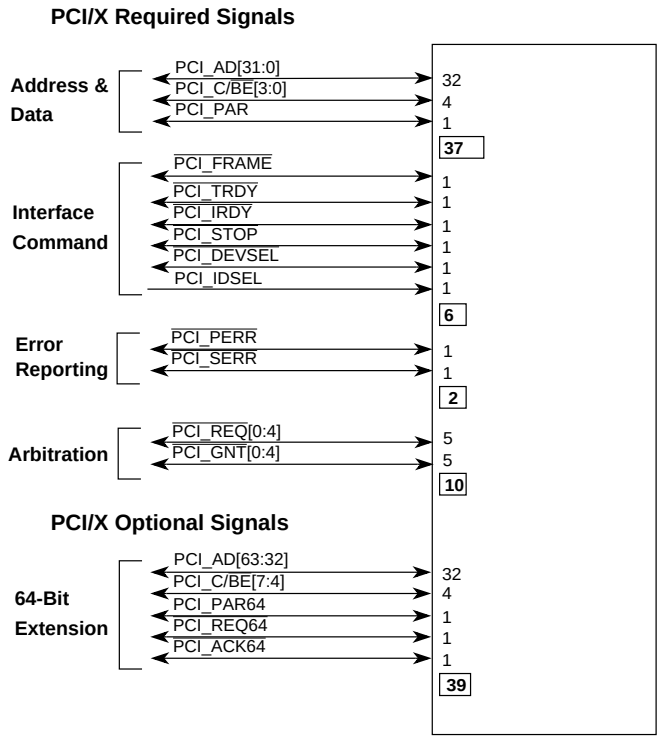


Figure 15-2. PCI/X Interface External Signals

Table 15-1 contains the detailed descriptions of the external PCI/PCI-X interface signals.

Table 15-1. 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 15-1. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_C/$\overline{\text{BE}}$[7:0]	I/O	Command/byte enable. The command/byte enable (PCI_C/$\overline{\text{BE}}$[7:0]) signals are both input and output signals on this PCI controller. The command encodings are as follows: PCI_C/$\overline{\text{BE}}$[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/$\overline{\text{BE}}$[7:0] define the bus command. Byte enables determine which byte lanes carry meaningful data. The PCI_C/$\overline{\text{BE}}$0 signal applies to the LSB.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bi-directional command/byte enable, these signals operate as described below.	
		State Meaning	Asserted/Negated—During the address phase, PCI_C/$\overline{\text{BE}}$[7:0] indicate the command that another master is sending. During the data phase, PCI_C/$\overline{\text{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
$\overline{\text{PCI_DEVSEL}}$	I/O	Device select. The device select ($\overline{\text{PCI_DEVSEL}}$) 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 15-1. 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.
		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 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.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_GNT[4:0]	O	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 *n*. Negated—Indicates that this PCI controller has not granted control of the PCI bus to agent *n*.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_IDSEL	I	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.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a

Table 15-1. 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 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 PCI_IRDY to insert a wait cycle when it cannot provide valid data to the target. During a read, this PCI controller negates 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 PCI_FRAME is asserted, indicates a wait cycle from another master. If PCI_FRAME is negated, indicates the PCI bus is idle.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
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/BE[3:0] signals during address and data phases. Negated—Indicates even parity across the PCI_AD[31:0] and PCI_C/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.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a

Table 15-1. 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/$\overline{BE}$[7:4] signals during address and data phases. Negated—Indicates even parity across the PCI_AD[63:32] and PCI_C/$\overline{BE}$[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
$\overline{PCI_PERR}$	I/O	PCI parity error. The PCI parity error ($\overline{PCI_PERR}$) 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 $\overline{PCI_PERR}$ on read operations; the PCI target drives $\overline{PCI_PERR}$ 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
$\overline{PCI_REQ}$[4:0]	I	PCI bus request**.** Input signals on this PCI controller when the arbiter is enabled. When the arbiter is disabled, $\overline{PCI_REQ}$[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 $\overline{PCI_REQ}$[n] input.	
		State Meaning	Asserted—Indicates that agent *n* is requesting control of the PCI bus to perform a transaction. Negated—Indicates that agent *n* 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 15-1. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{PCI_REQ64}}$	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
$\overline{\text{PCI_SERR}}$	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 15-1. 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.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a
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 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 PCI_TRDY to insert a wait cycle when it cannot provide valid data to the initiator. During a write, this PCI controller negates 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.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specs Rev 2.2 or by PCI-X Addendum Rev 1.0a

15.3 Memory Map/Register Definitions

The PCI/PCI-X block of the MPC8560 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 MPC8560. These registers are described in Section 15.3.1, “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 15.3.2, “PCI/PCI-X Configuration Header,” and its subsections.

15.3.1 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 15-2 lists the memory-mapped registers.

Table 15-2. 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	15.3.2.1/15-34
0x0_8004	PCI Configuration data access (PCI CFG_DATA)	R/W	0x0000_0000	15.3.2.1/15-34
0x0_8008	PCI/X Int_Ack access	R	0x0000_0000	15.3.1.4/15-33
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	15.3.1.1.1/15-17
0x0_8C04	PCI/X Outbound window 0 (default) translation extended address register	R/W	0x0000_0000	15.3.1.1.2/15-18
0x0_8C08	PCI/X outbound window 0 (default) base address register	R/W	0x0000_0000	15.3.1.1.3/15-19
0x0_8C0C	Reserved	—	—	
0x0_8C10	PCI/X outbound window 0 (default) attributes register	R/W	0x8004_401F	15.3.1.1.4/15-19
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	15.3.1.1.1/15-17
0x0_8C24	PCI/X Outbound window 1 translation extended address register	R/W	0x0000_0000	15.3.1.1.2/15-18
0x0_8C28	PCI/X Outbound window 1 base address register	R/W	0x0000_0000	15.3.1.1.3/15-19
0x0_8C2C	Reserved	—	—	

Table 15-2. 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	15.3.1.1.4/15-19
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	15.3.1.1.1/15-17
0x0_8C44	PCI/X Outbound window 2 translation extended address register	R/W	0x0000_0000	15.3.1.1.2/15-18
0x0_8C48	PCI/X Outbound window 2 base address register	R/W	0x0000_0000	15.3.1.1.3/15-19
0x0_8C4C	Reserved	—	—	
0x0_8C50	PCI/X Outbound window 2 attributes register	R/W	0x0000_0000	15.3.1.1.4/15-19
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	15.3.1.1.1/15-17
0x0_8C64	PCI/X Outbound window 3 translation extended address register	R/W	0x0000_0000	15.3.1.1.2/15-18
0x0_8C68	PCI/X Outbound window 3 base address register	R/W	0x0000_0000	15.3.1.1.3/15-19
0x0_8C6C	Reserved	—	—	
0x0_8C70	PCI/X Outbound window 3 attributes register	R/W	0x0000_0000	15.3.1.1.4/15-19
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	15.3.1.1.1/15-17
0x0_8C84	PCI/X Outbound window 4 translation extended address register	R/W	0x0000_0000	15.3.1.1.2/15-18
0x0_8C88	PCI/X Outbound window 4 base address register	R/W	0x0000_0000	15.3.1.1.3/15-19
0x0_8C8C	Reserved	—	—	
0x0_8C90	PCI/X Outbound window 4 attributes register	R/W	0x0000_0000	15.3.1.1.4/15-19
0x0_8C94	Reserved	—	—	
0x0_8C98	Reserved	—	—	
0x0_8C9C	Reserved	—	—	

Table 15-2. 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	15.3.1.2.1/15-22
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	15.3.1.2.2/15-23
0x0_8DAC	Reserved	—	—	
0x0_8DB0	PCI/X Inbound window 3 attributes register	R/W	0x0000_0000	15.3.1.2.4/15-24
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	15.3.1.2.1/15-22
0x0_8DC4	Reserved	—	—	
0x0_8DC8	PCI/X Inbound window 2 base address register	R/W	0x0000_0000	15.3.1.2.2/15-23
0x0_8DCC	PCI/X Inbound window 2 base extended address register	R/W	0x0000_0000	15.3.1.2.3/15-23
0x0_8DD0	PCI/X inbound window 2 attributes register	R/W	0x0000_0000	15.3.1.2.4/15-24
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	15.3.1.2.1/15-22
0x0_8DE4	Reserved	—	—	
0x0_8DE8	PCI/X Inbound window 1 base address register	R/W	0x0000_0000	15.3.1.2.2/15-23
0x0_8DEC	Reserved	—	—	
0x0_8DF0	PCI/X Inbound window 1 attributes register	R/W	0x0000_0000	15.3.1.2.4/15-24
0x0_8DF4	Reserved	—	—	
0x0_8DF8	Reserved	—	—	
0x0_8DFC	Reserved	—	—	
PCI/X Error Management Registers				
0x0_8E00	PCI/X error detect register	Special	0x0000_0000	15.3.1.3/15-26
0x0_8E04	PCI/X error capture disabled register	R/W	0x0000_0000	15.3.1.3/15-26

Table 15-2. PCI/X Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8E08	PCI/X error enable register	R/W	0x0000_0000	15.3.1.3/15-26
0x0_8E0C	PCI/X error attributes capture register	R/W	0x0000_0000	15.3.1.3/15-26
0x0_8E10	PCI/X error address capture register	R/W	0x0000_0000	15.3.1.3/15-26
0x0_8E14	PCI/X error extended address capture register	R/W	0x0000_0000	15.3.1.3/15-26
0x0_8E18	PCI/X error data low capture register	R/W	0x0000_0000	15.3.1.3/15-26
0x0_8E1C	PCI/X error data high capture register	R/W	0x0000_0000	15.3.1.3/15-26
0x0_8E20	PCI/X gasket timer register	R/W	0x0000_0000	15.3.1.3/15-26
0x0_8E24–0x0_8EFC	Reserved	—	—	
0x0_8F00–0x0_8FFC	Reserved for debug	—	—	

15.3.1.1 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 MPC8560 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.

15.3.1.1.1 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 15-3 shows the field descriptions of the PCI/X outbound translation address registers.

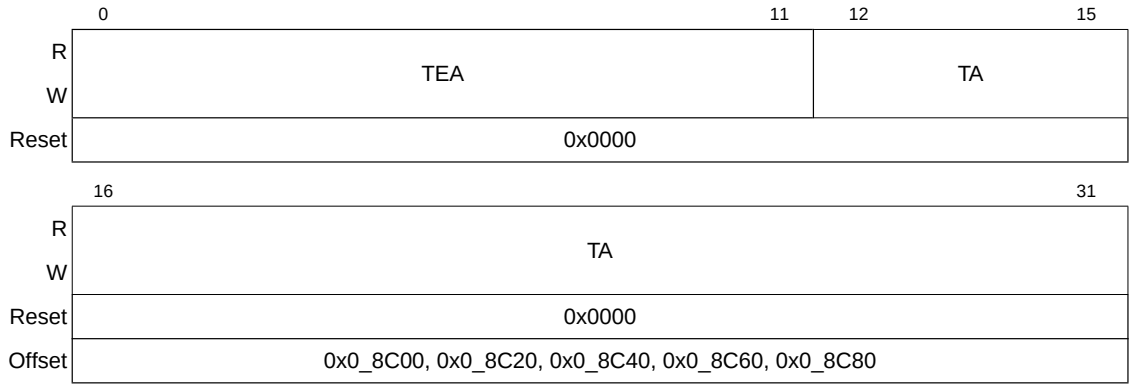


Figure 15-3. PCI/X Outbound Translation Address Register

Table 15-3. 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.

15.3.1.1.2 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 15-4 shows the field descriptions of the PCI/X outbound translation extended address registers.

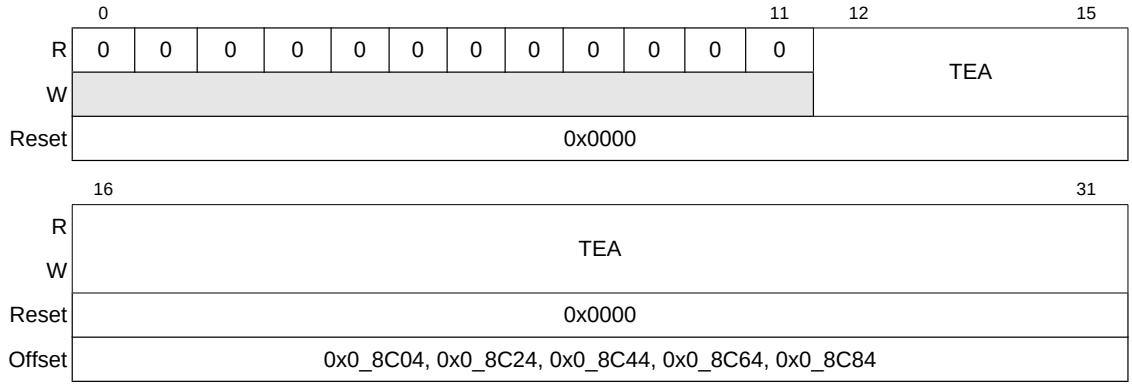


Figure 15-4. PCI/X Outbound Translation Extended Address Register

Table 15-4. 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

15.3.1.1.3 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.

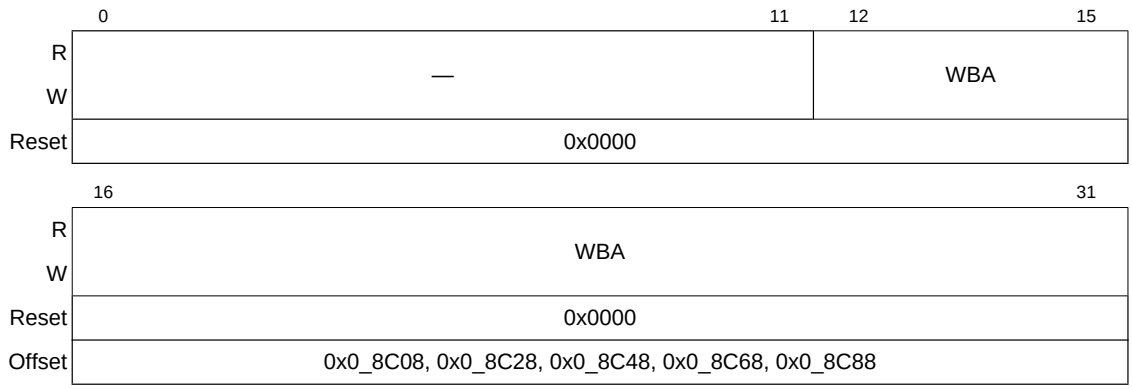


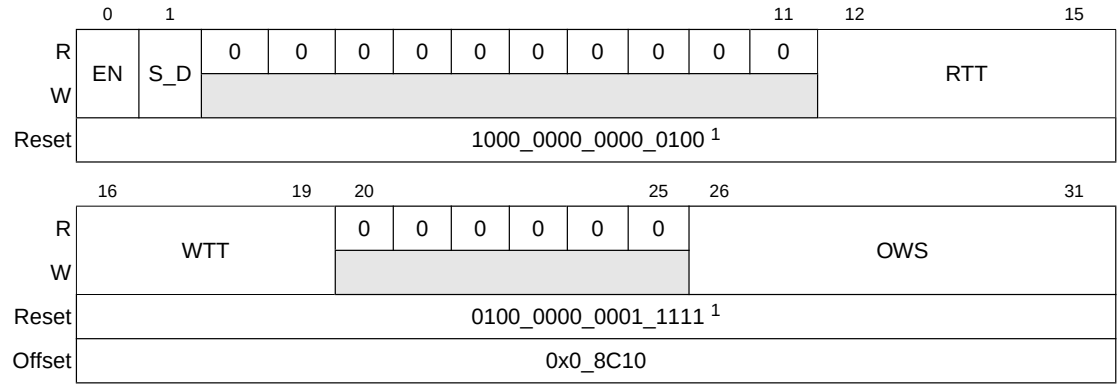
Figure 15-5. PCI/X Outbound Window Base Address Register

Table 15-5. 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.

15.3.1.1.4 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 15-6 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).



1 The default window is enabled, configured for memory read and memory write, and set to an OWS size of 4 Gbytes.

Figure 15-6. PCI/X Outbound Window Attributes Register 0 (Default)

Figure 15-7 shows the reset values for POWAR1–POWAR4.

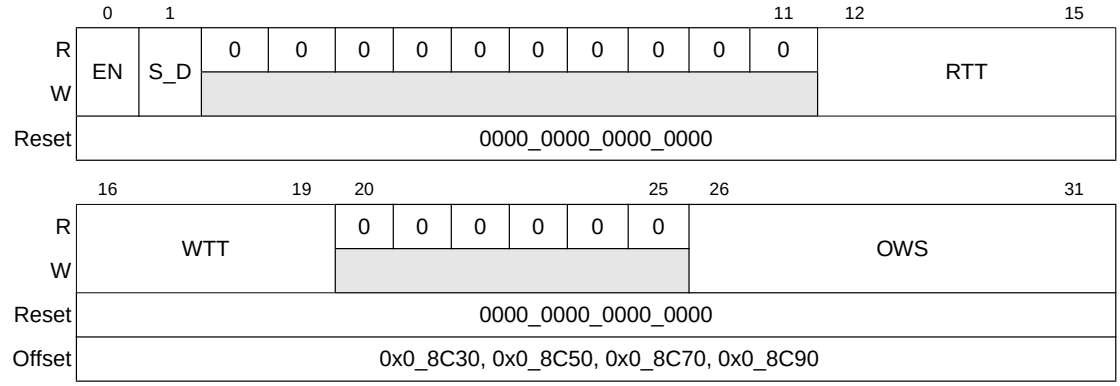


Figure 15-7. PCI/X Outbound Window Attributes Registers 1–4

Table 15-6. 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 15-6. POWAR*n* 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.

15.3.1.2 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 MPC8560. 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 15-21]. 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 15.3.2.1, “Accessing PCI/X Configuration Header Registers,’ and Section 15.3.2.9, “PCI Base Address Registers.”

15.3.1.2.1 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.

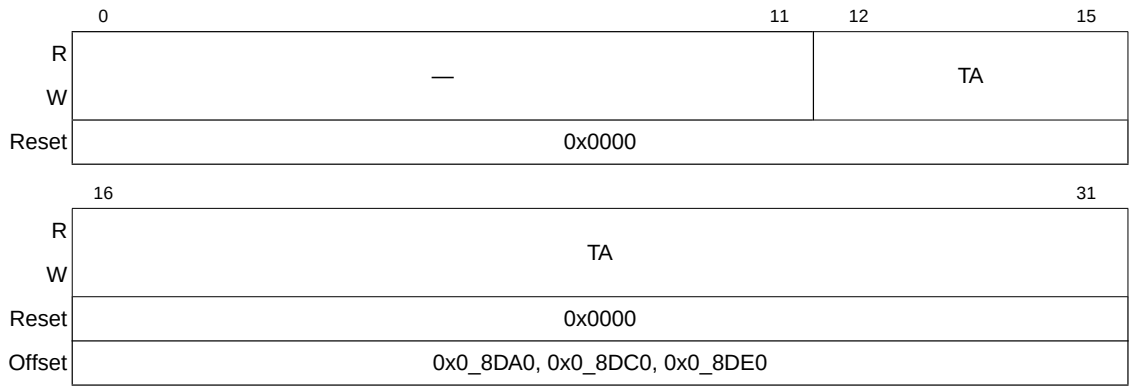


Figure 15-8. PCI/X Inbound Translation Address Register

Table 15-7. 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.

15.3.1.2.2 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 MPC8560. 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.

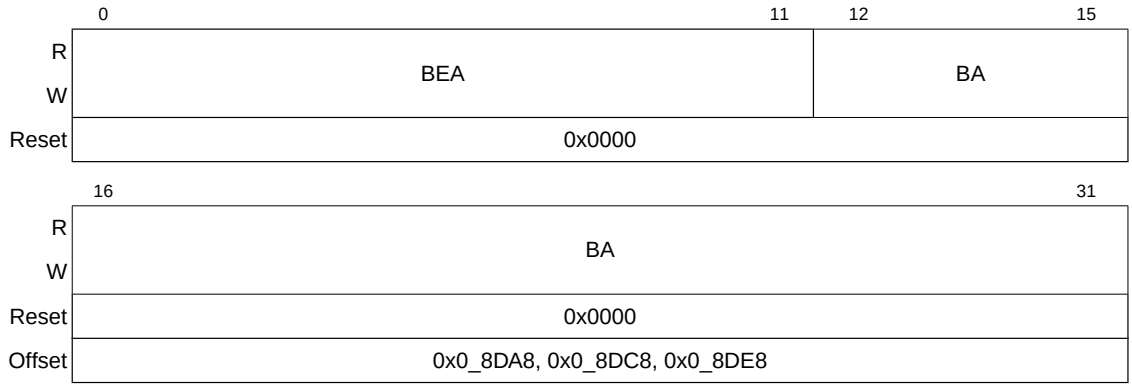


Figure 15-9. PCI/X Inbound Window Base Address Register

Table 15-8. 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.

15.3.1.2.3 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.

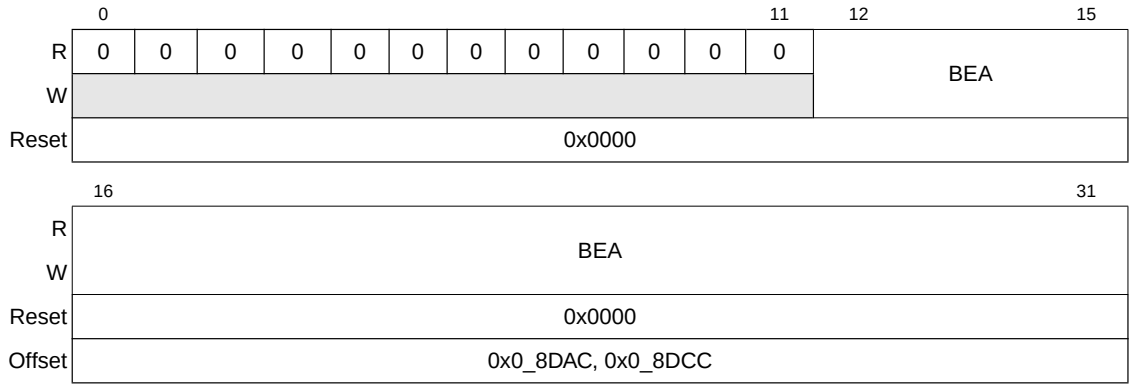


Figure 15-10. PCI/X Inbound Window Base Extended Address Register

Table 15-9. 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.

15.3.1.2.4 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 15-11. PCI/X Inbound Window Attributes Register

Table 15-10. 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 15-10. 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.

15.3.1.3 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.

15.3.1.3.1 PCI/X Error Detect Register

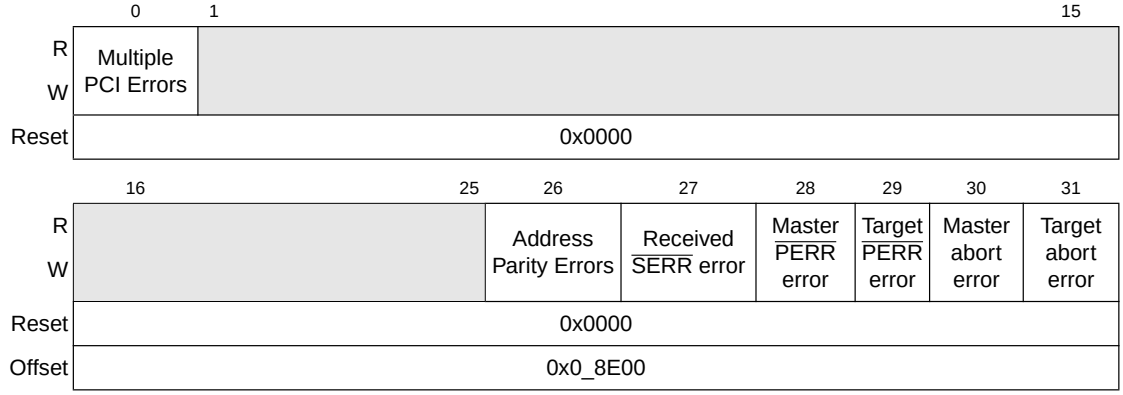


Figure 15-12. PCI/X Error Detect Register

Table 15-11. 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)

15.3.1.3.2 PCI/X Error Capture Disable Register

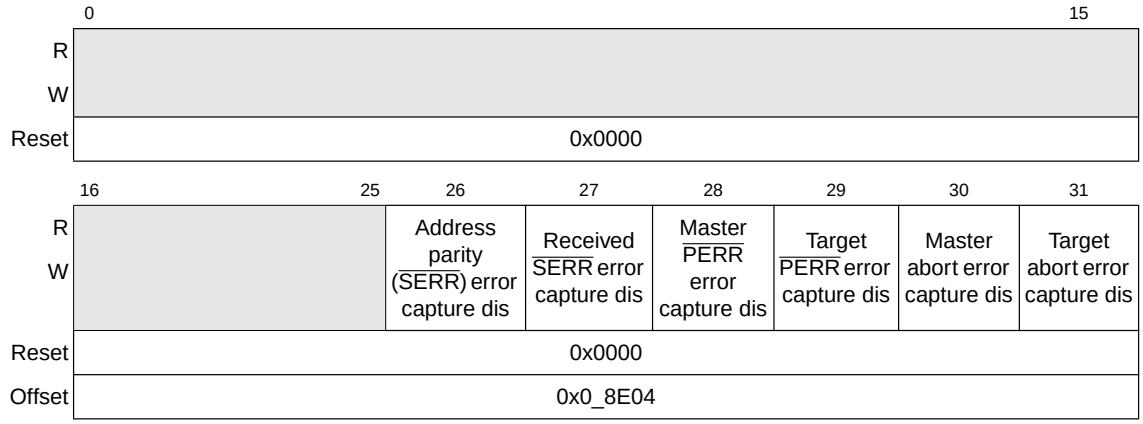


Figure 15-13. PCI/X Error Capture Disable Register

Table 15-12. 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

15.3.1.3.3 PCI/X Error Enable Register

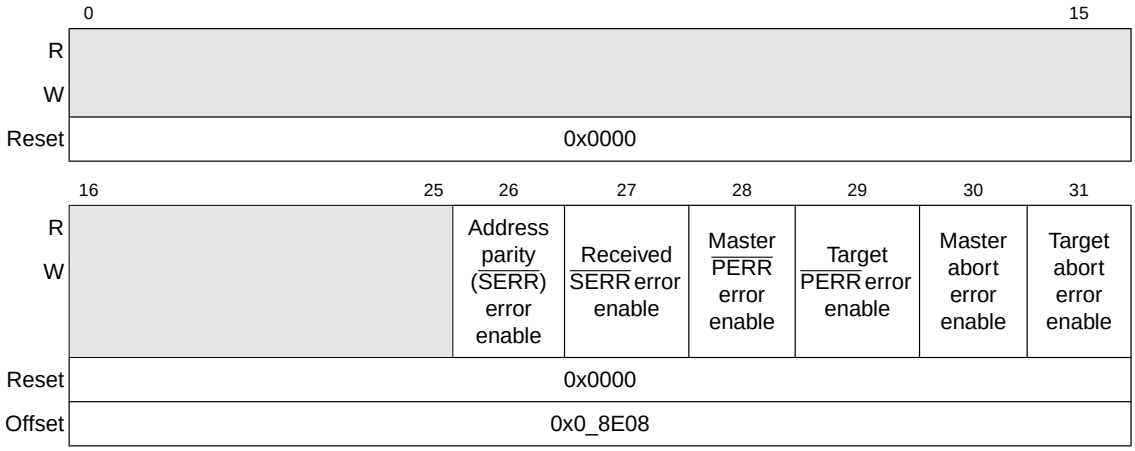


Figure 15-14. PCI/X Error Enable Register

Table 15-13. Bit Settings for PCI/X Error Enable Register

Bits	Name	Description
0–25	—	Reserved
26	Address parity ($\overline{\text{SERR}}$) error enable	Enable reporting address parity errors
27	Received $\overline{\text{SERR}}$ error enable	Enable reporting received $\overline{\text{SERR}}$ errors
28	Master $\overline{\text{PERR}}$ error enable	Enable reporting master $\overline{\text{PERR}}$ errors
29	Target $\overline{\text{PERR}}$ error enable	Enable reporting target $\overline{\text{PERR}}$ errors
30	Master abort error enable	Enable reporting master abort errors
31	Target abort error enable	Enable reporting target abort errors

15.3.1.3.4 PCI/X Error Attributes Capture Register

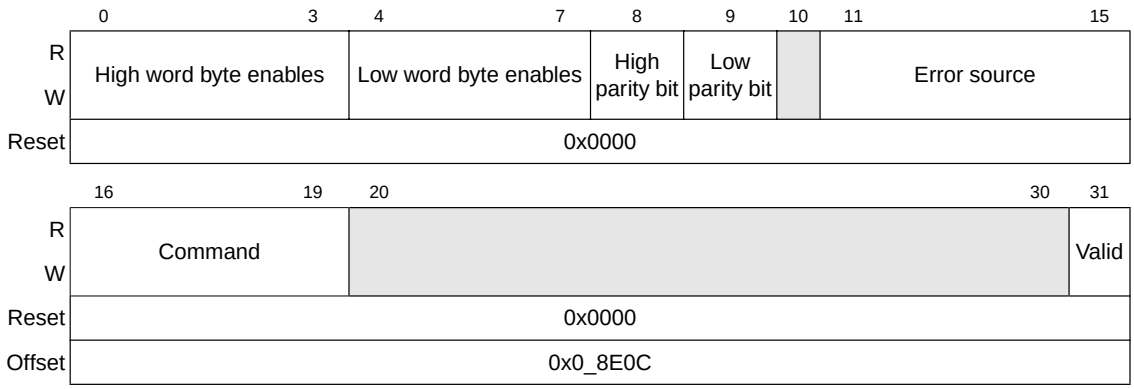


Figure 15-15. PCI/X Error Attributes Capture Register

Table 15-14. 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 - CPM 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

15.3.1.3.5 PCI/X Error Address Capture Register

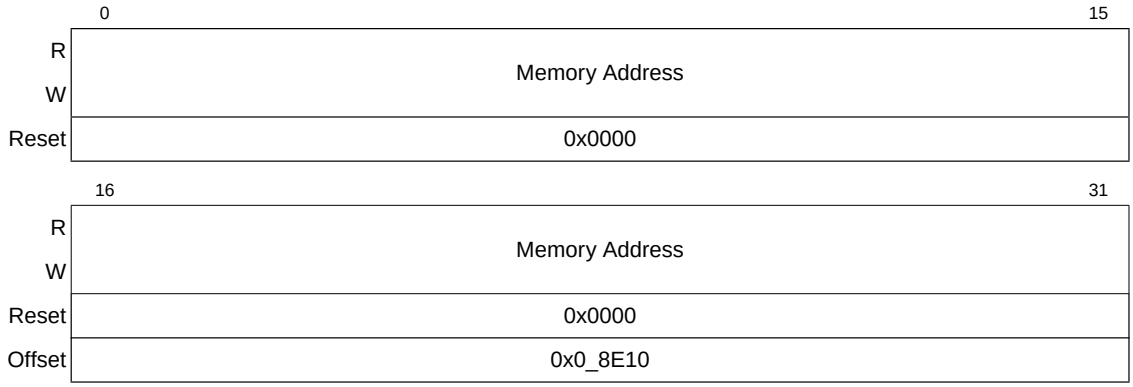


Figure 15-16. PCI/X Error Address Capture Register

Table 15-15. Bit Settings for PCI/X Error Address Capture Register

Bits	Name	Description
0–31	Memory address	Memory transaction address

15.3.1.3.6 PCI/X Error Extended Address Capture Register

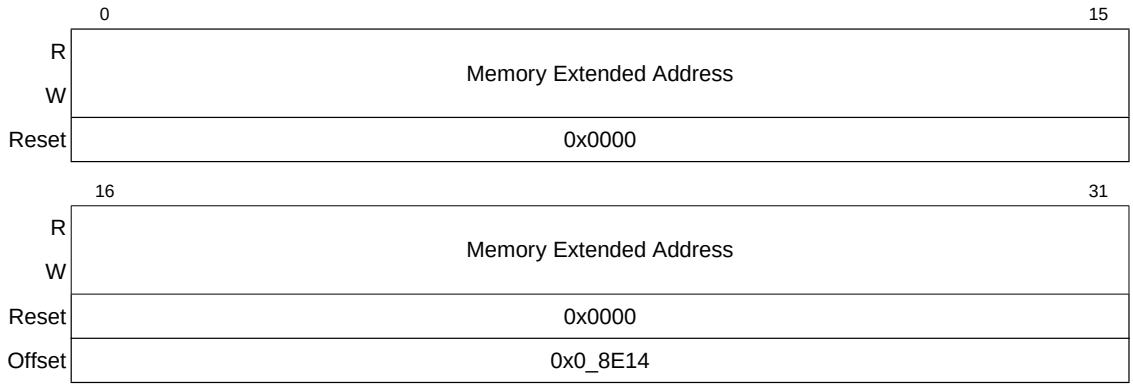


Figure 15-17. PCI/X Error Extended Address Capture Register

Table 15-16. Bit Settings for PCI/X Error Extended Address Capture Register

Bits	Name	Description
0–31	Memory extended address	Memory transaction extended address

15.3.1.3.7 PCI/X Error Data Low Capture Register

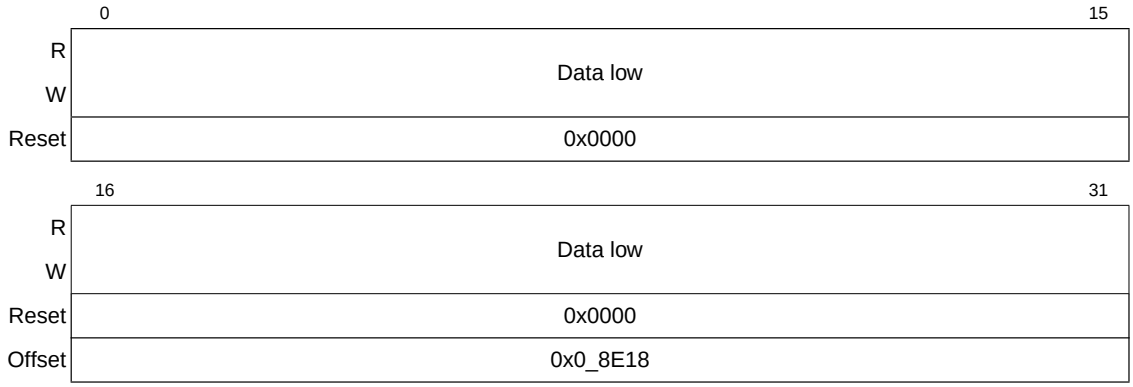


Figure 15-18. PCI/X Error Data Low Capture Register

Table 15-17. Bit Settings for PCI/X Error Data Low Capture Register

Bits	Name	Description
0–31	Data low	Least significant PCI bus data word

15.3.1.3.8 PCI/X Error Data High Capture Register

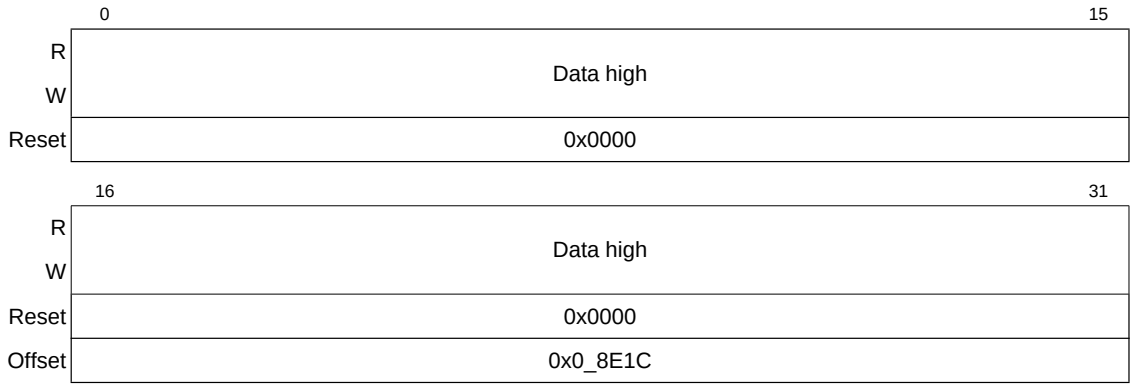


Figure 15-19. PCI/X Error Data High Capture Register

Table 15-18. 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)

15.3.1.4 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.

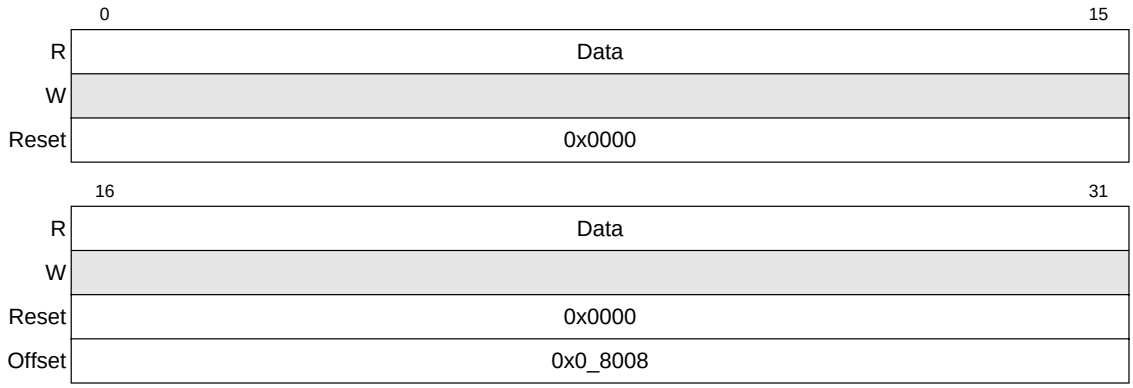


Figure 15-20. PCI Interrupt Acknowledge Register

15.3.2 PCI/PCI-X Configuration Header

Figure 15-21 lists the common PCI/PCI-X configuration header as implemented by the MPC8560.

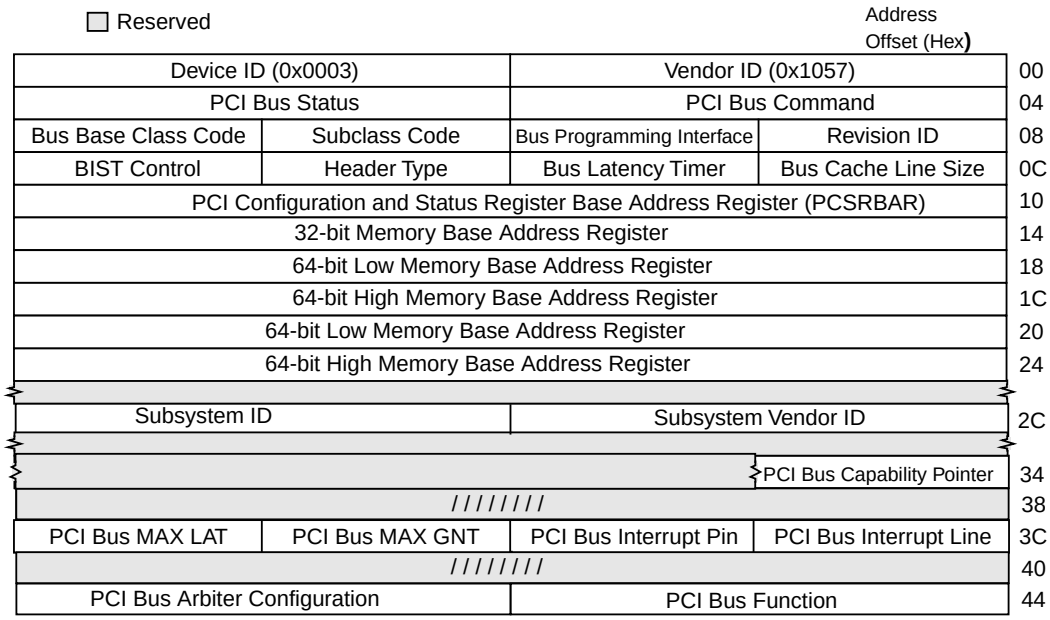


Figure 15-21. MPC8560 Common PCI and PCI-X Configuration Header

Figure 15-22 lists the additional PCI-X configuration registers.

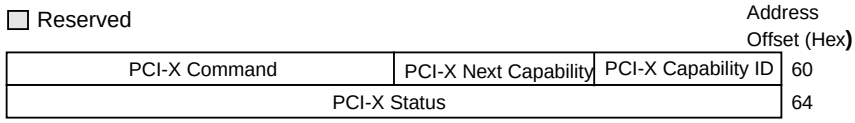


Figure 15-22. PCI-X Additional Configuration Registers

15.3.2.1 Accessing PCI/X Configuration Header Registers

The PCI/X configuration registers, shown in Figure 15-21 and Figure 15-22, 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 15-23, is described in Table 15-19. 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.

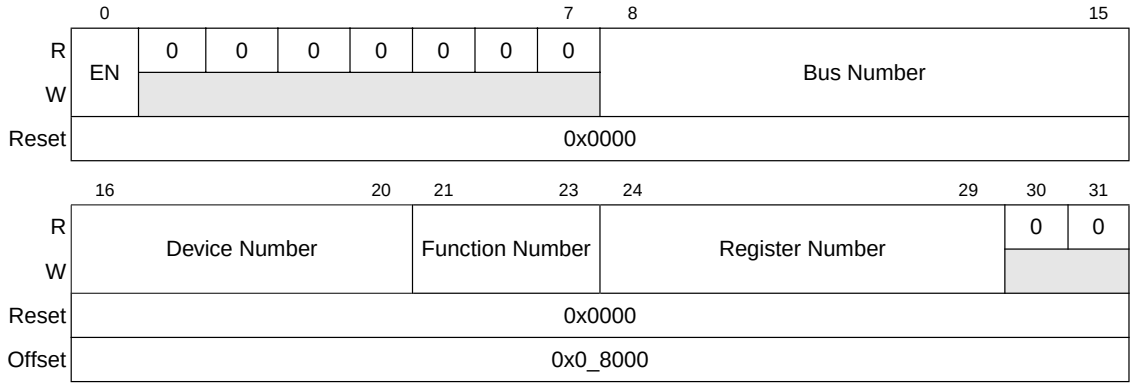


Figure 15-23. PCI/PCI-X Configuration Header Register

Table 15-19. 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 MPC8560 itself.

The CFG_DATA register, shown in Figure 15-23, is a 32-bit, data-only register and is described in Table 15-20.

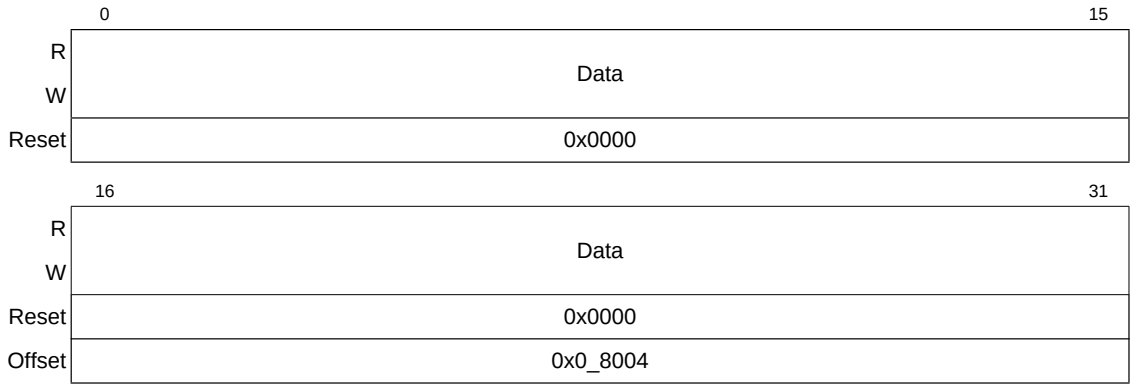


Figure 15-24. PCI CFG_DATA Register

Table 15-20. 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.

15.3.2.2 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.

15.3.2.3 PCI Bus Command Register—Offset 0x04

Table 15-21. 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 15-22 describes the bits of the PCI bus command register.

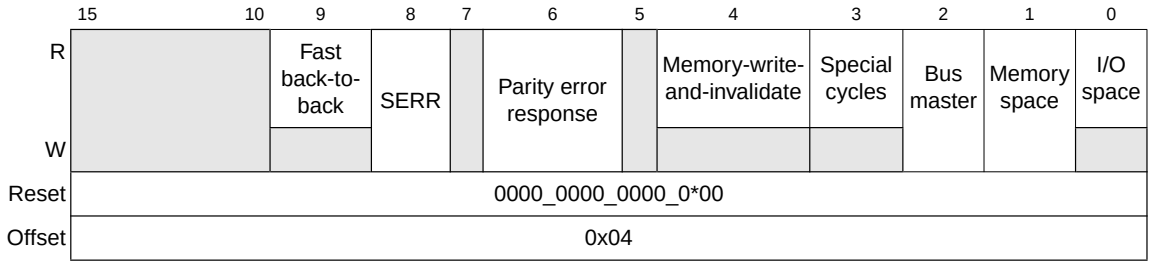


Figure 15-25. PCI Bus Command Register

Table 15-22. 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.

15.3.2.4 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 15-23. 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.

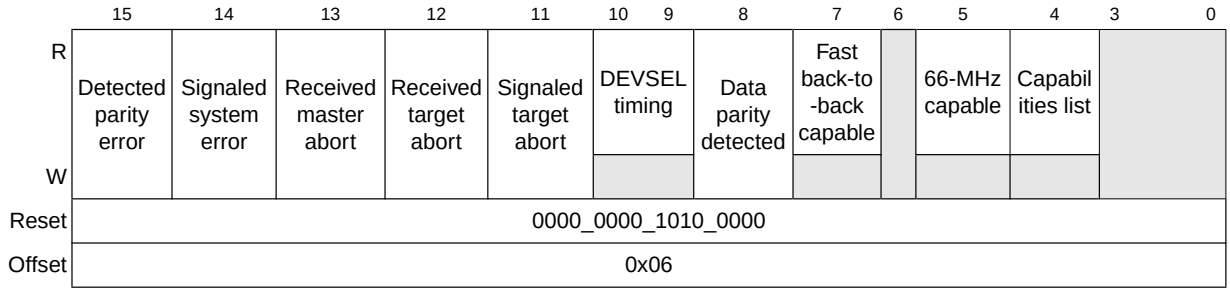


Figure 15-26. PCI Bus Status Register

Table 15-23. 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 $\overline{\text{PCI_SERR}}$.
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 15-23. 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

15.3.2.5 PCI Bus Programming Interface Register—Offset 0x09

Table 15-24 describes the PCI Bus programming interface register (PIR).

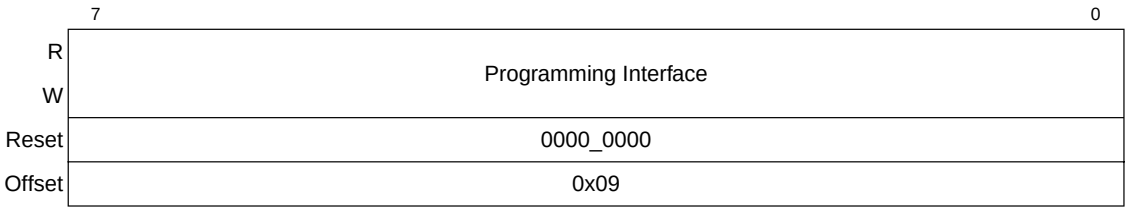


Figure 15-27. PCI Bus Programming Interface Register

Table 15-24. 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

15.3.2.6 PCI Bus Base Class Code Register—Offset 0x0B

Table 15-25 describes the PCI bus base code register (PBCCR).

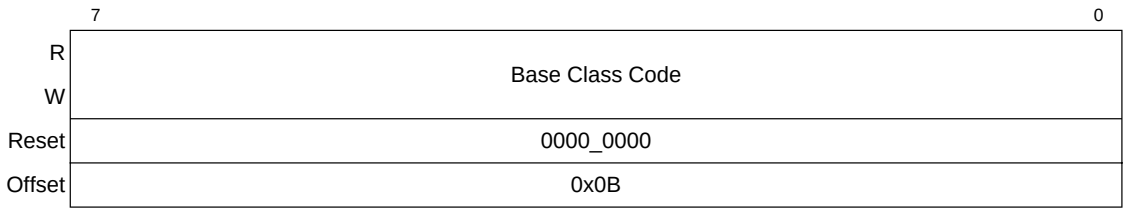


Figure 15-28. PCI Bus Base Class Code Register

Table 15-25. 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.

15.3.2.7 PCI Bus Cache Line Size Register—Offset 0x0C

Table 15-26 describes the PCI bus cache line size register (PCLSR).

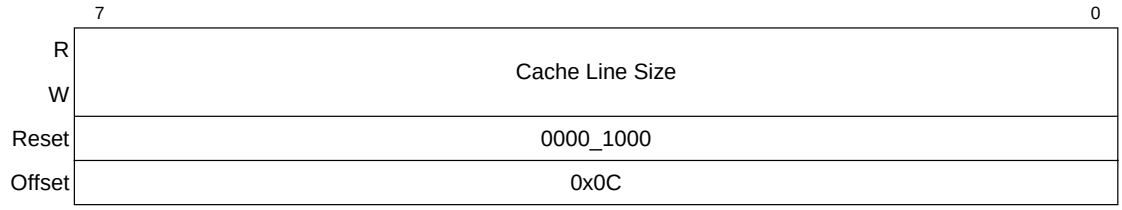


Figure 15-29. PCI Bus Cache Line Size Register

Table 15-26. 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.

15.3.2.8 PCI Bus Latency Timer Register—0x0D

Table 15-27 describes the PCI latency timer register (PLTR).

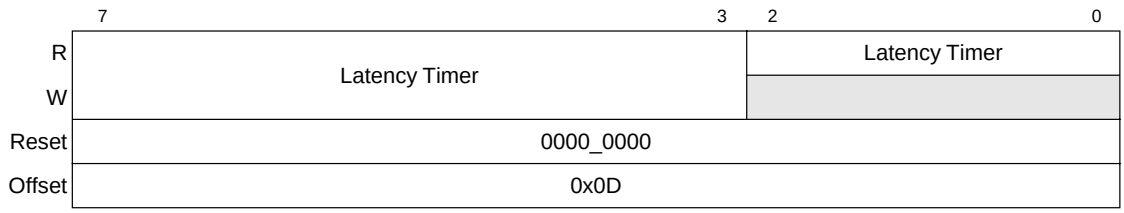


Figure 15-30. PCI Bus Latency Timer Register

Table 15-27. 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.

15.3.2.9 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 15.3.1.2, “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.

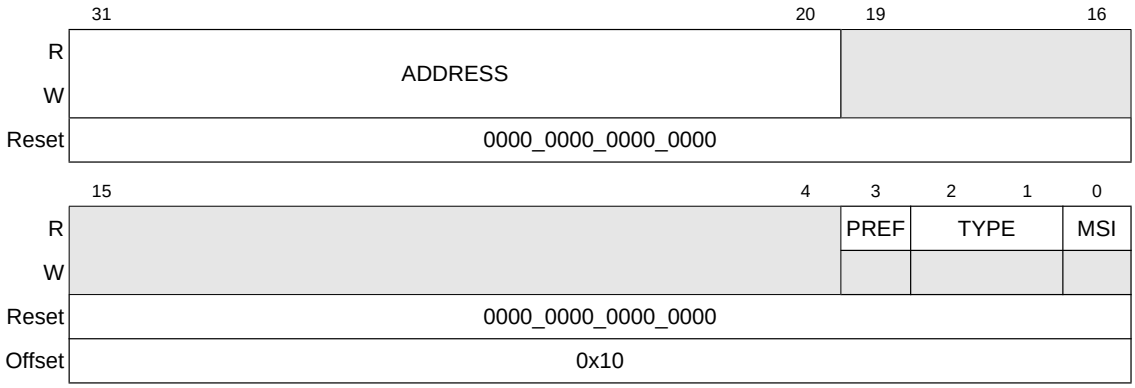


Figure 15-31. PCI Configuration and Status Register Base Address Register (PCSRBAR)

Table 15-28. 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 15-28. Bit Settings for PCSRBAR

Bits	Name	Description
2–1	TYPE	Type. Locate anywhere in 32-bit address space.
0	MSI	Memory space indicator

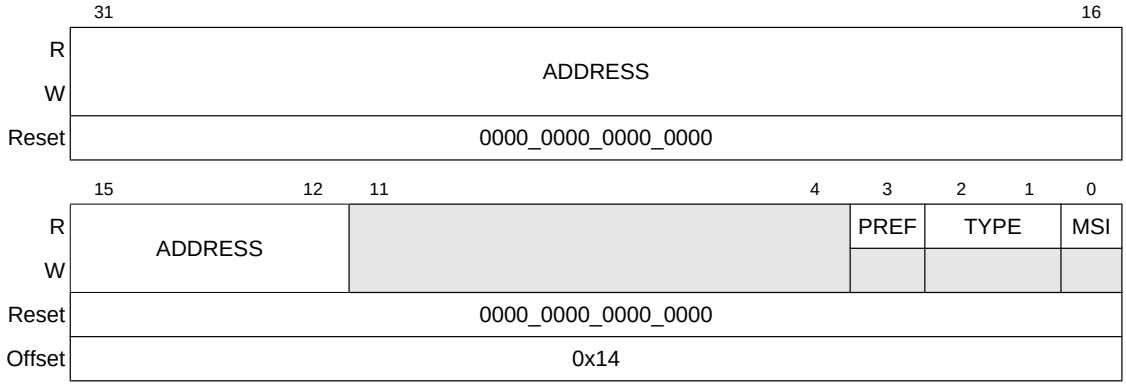


Figure 15-32. 32–Bit Memory Base Address Register

Table 15-29. 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.

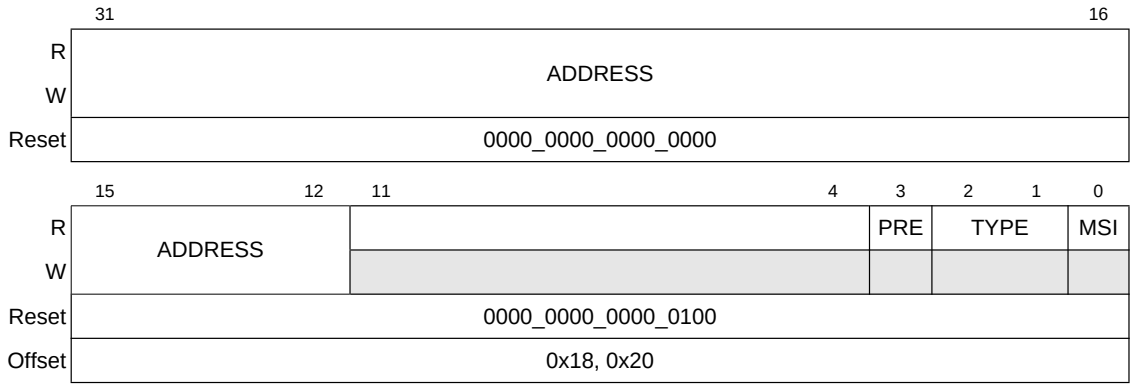


Figure 15-33. 64-bit Low Memory Base Address Register

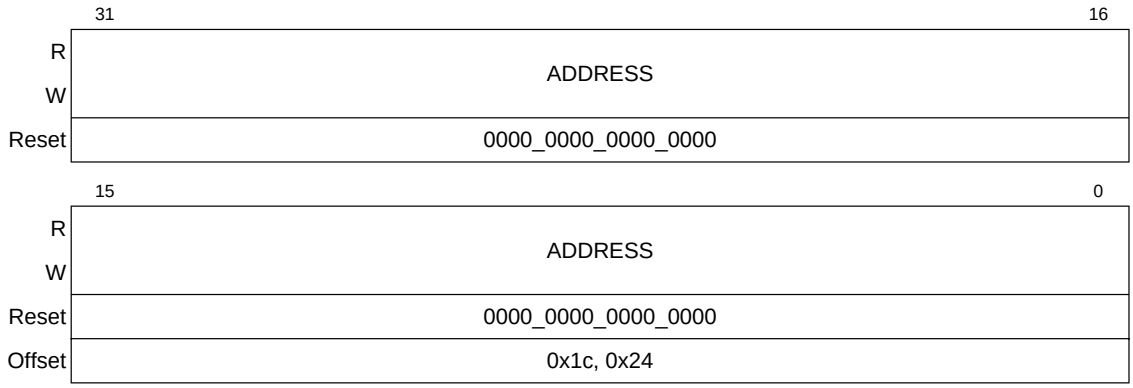


Figure 15-34. 64–bit High Memory Base Address Register

Table 15-30. 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 15-31. 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.

15.3.2.10 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 15-32.

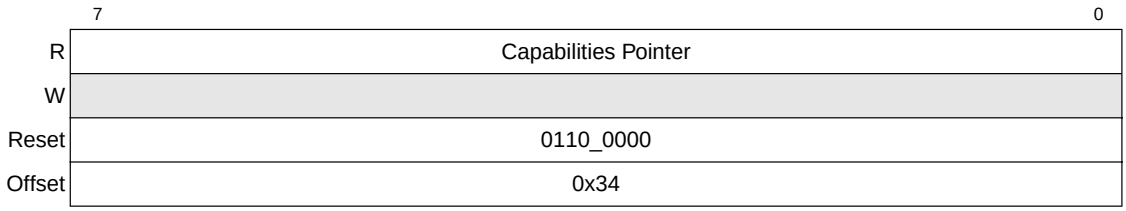


Figure 15-35. PCI Bus Capabilities Pointer Register

Table 15-32. 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.

15.3.2.11 PCI Bus Interrupt Line Register

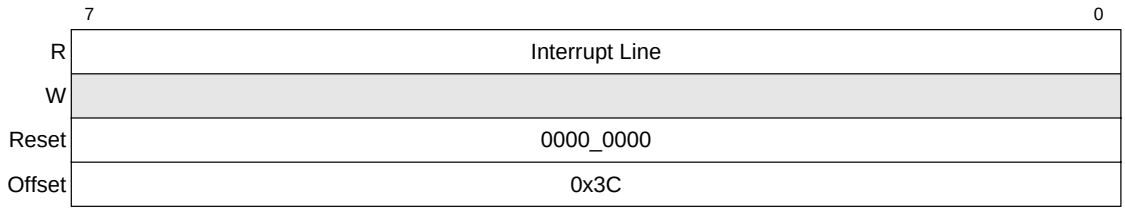


Figure 15-36. PCI Bus Interrupt Line Register

Table 15-33. Bit Settings for PCI Bus Interrupt Line Register—0x3C

Bits	Name	Description
7–0	Interrupt Line	Reserved

15.3.2.12 PCI Bus Interrupt Pin Register

The MPC8560 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 MPC8560 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.

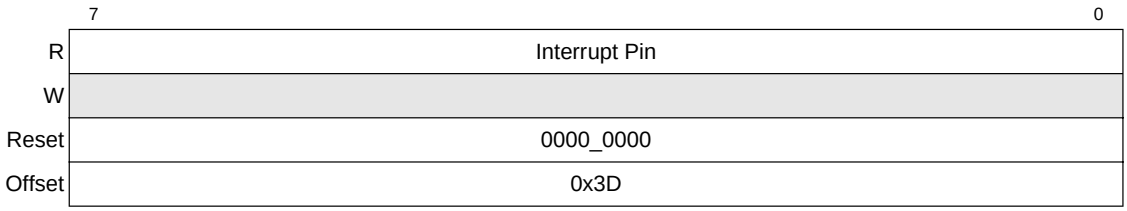


Figure 15-37. PCI Bus Interrupt Pin Register

Table 15-34. Bit Settings for PCI Bus Interrupt Pin Register—0x3D

Bits	Name	Description
7–0	Interrupt Pin	No PCI_INT pin selected

15.3.2.13 PCI Bus Maximum Grant (MAX GNT) Register

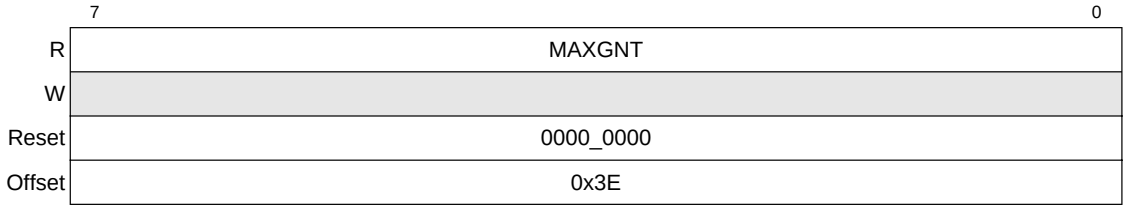


Figure 15-38. PCI Bus Maximum Grant Register

Table 15-35. 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)

15.3.2.14 PCI Bus Maximum Latency (MAX LAT) Register

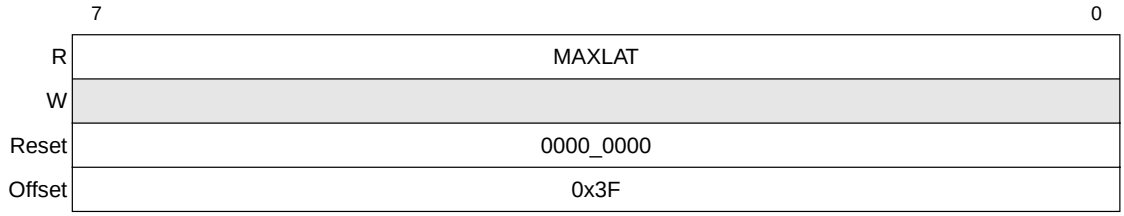


Figure 15-39. PCI Bus Maximum Latency Register

Table 15-36. 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)

15.3.2.15 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	PAH
W																
Reset	0000_0000_00*0_*00*															
Offset	0x44															

Figure 15-40. PCI Bus Function Register

Table 15-37. 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

15.3.2.16 PCI Bus Arbiter Configuration Register (PBACR)

The PCI bus arbiter configuration register is used to determine the configuration of the PCI bus arbiter.

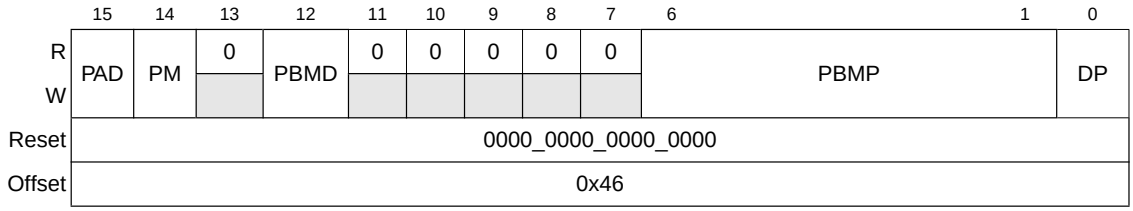


Figure 15-41. PCI Bus Arbiter Configuration Register

Table 15-38. 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 $\overline{\text{PCI_GNT2}}$ 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 *n* is low priority. 1 Master *n* 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.

15.3.2.17 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*.

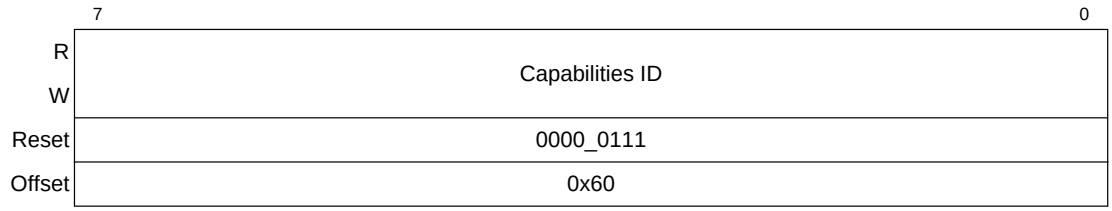


Figure 15-42. PCI-X Next Capabilities ID Register

Table 15-39. 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

15.3.2.18 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 15-40.

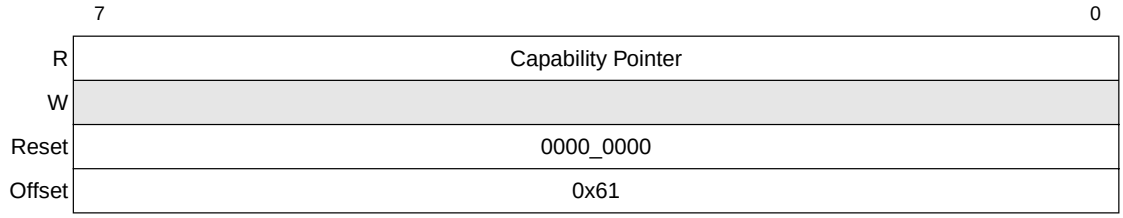


Figure 15-43. PCI-X Capability Pointer Register

Table 15-40. 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.

15.3.2.19 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 15-41 describes the bits of the PCI-X command register.

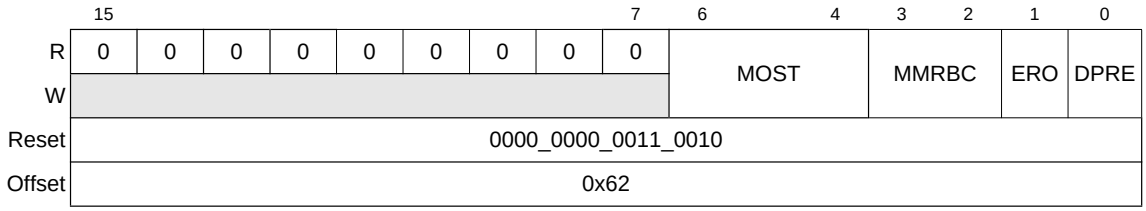


Figure 15-44. PCI-X Command Register

Table 15-41. 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)

15.3.2.20 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 15-42. Only 4-byte accesses to address offset 0x64 are allowed.

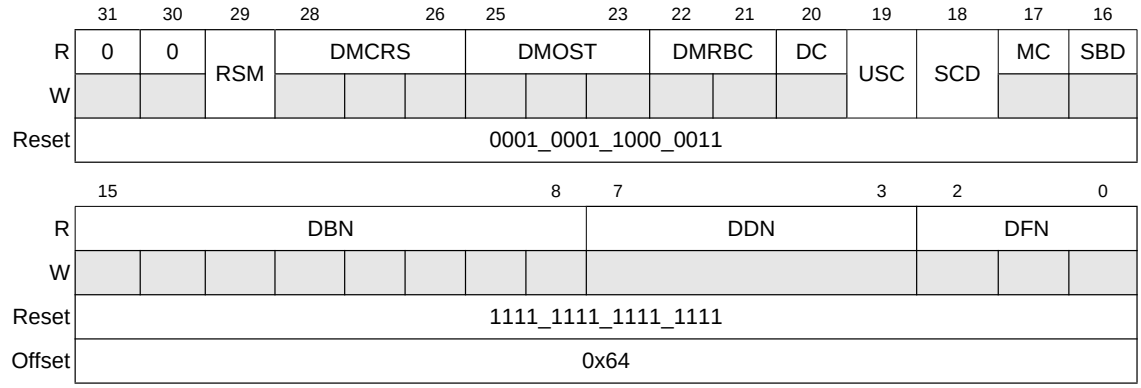


Figure 15-45. PCI-X Status Register

Table 15-42. 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 15-42. 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)

15.4 Functional Description

This section describes the functionality of the PCI/PCI-X interface.

15.4.1 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 MPC8560 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 MPC8560 that governs processor, DMA, or RapidIO accesses to the PCI interface. If the on-chip PCI arbiter is disabled, the MPC8560 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 MPC8560.

15.4.1.1 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 MPC8560, 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 MPC8560 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 15-46, 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 15-46, the grant sequence (with all devices, except device 4 requesting the bus and device 3 being the current master) is 0, 2, MPC8560, 0, 2, 1, 0, 2, 3, …, and repeating. If device 2 is not requesting the bus, the grant sequence is 0, MPC8560, 0, 1, 0, 3, …, and repeating. If device 2 requests the bus when device 0 is conducting a transaction and the MPC8560 has the next grant, the MPC8560 has its grant removed and device 2 is awarded the grant since device 2 is higher priority than the MPC8560 when device 0 has the bus.

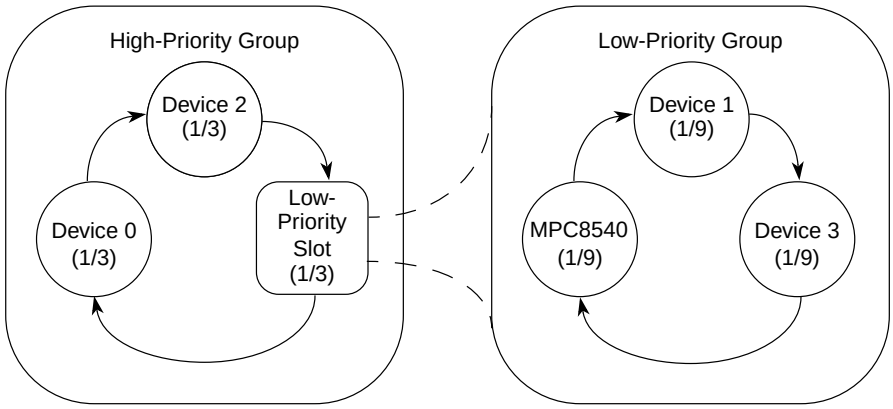


Figure 15-46. PCI Arbitration Example

15.4.1.2 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/$\overline{\text{BE}}$[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 MPC8560.

15.4.1.3 Broken Master Lock-Out

The PCI bus arbiter on the MPC8560 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 $\overline{\text{PCI_FRAME}}$ 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.

15.4.1.4 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 MPC8560 when its clocks are stopped, the MPC8560 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 MPC8560. In nap and doze power-saving modes, the arbiter continues to operate allowing other PCI devices to run transactions.

15.4.2 PCI Bus Protocol

This section provides a general description of the PCI bus protocol. Specific PCI bus transactions are described in Section 15.4.2.7, “PCI Bus Transactions.” Refer to Figure 15-47, Figure 15-48, Figure 15-49, and Figure 15-50 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 *MPC8560 Hardware Specifications* for specific setup and hold times.

15.4.2.1 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 PCI_IRDY and PCI_TRDY are asserted. Wait cycles may be inserted in a data phase by the initiator (by negating PCI_IRDY) or by the target (by negating PCI_TRDY).

Once an initiator has asserted PCI_IRDY, it cannot change PCI_IRDY or PCI_FRAME until the current data phase completes regardless of the state of PCI_TRDY. Once a target has asserted PCI_TRDY or PCI_STOP, it cannot change PCI_DEVSEL, PCI_TRDY, or 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), PCI_FRAME is negated and PCI_IRDY is asserted (or kept asserted), indicating the initiator is ready. After the target indicates the final data transfer (by asserting PCI_TRDY), the PCI bus may return to the idle state (both PCI_FRAME and 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.

15.4.2.2 PCI Bus Commands

A PCI bus command is encoded in the PCI_C/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 15-43 describes the PCI bus commands implemented by the MPC8560.

Table 15-43. PCI Bus Commands

PCI_C/ BE[3:0]	PCI Bus Command	MPC8560 Supports as an Initiator	MPC8560 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 15.4.2.12.1, “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 15.4.2.12.2, “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 15-43. PCI Bus Commands (continued)

PCI_C/ $\overline{BE}$[3:0]	PCI Bus Command	MPC8560 Supports as an Initiator	MPC8560 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 MPC8560 (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 15.4.2.11, “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 15.4.2.11, “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 MPC8560 does not respond to these commands.

15.4.2.3 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 15.3.1, “PCI/PCI-X Memory Mapped Registers.” Access to the PCI configuration space is described in Section 15.4.2.11, “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 15.4.2.4, “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.

15.4.2.3.1 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 15-44. The other two PCI_AD[1:0] possibilities (0b01 and 0b11) are reserved. As an initiator, the MPC8560 always encodes PCI_AD[1:0] = 0b00 for PCI memory space accesses. As a target, the MPC8560 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 15.4.2.8.2, “Target-Initiated Termination,” for more information on target disconnect conditions.

Table 15-44. Supported Combinations of PCI_AD[1:0]

PCI_AD[1:0]		MPC8560 as Target		MPC8560 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 (PCI_AD[1:0] = 0b10) reads, the critical memory address is decoded using 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 MPC8560 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 PCI_AD[1:0] = 0b10. That is, the MPC8560 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.

15.4.2.3.2 I/O Space Addressing

For PCI I/O accesses, 32 address signals (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 15.4.2.8.2, “Target-Initiated Termination,” for more information.

15.4.2.3.3 Configuration Space Addressing

PCI supports two types of configuration accesses that use different formats for the 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 (PCI_AD[1:0] = 0b00) or type 1 (PCI_AD[1:0] = 0b01). Both address formats identify a specific device and a specific configuration register for that device. See Section 15.4.2.11, “Configuration Cycles,” for descriptions of the two formats.

15.4.2.4 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 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 MPC8560 is hardwired for fast device select timing (PCI status register [10–9] = 0b00). Therefore, when the MPC8560 is the target of a transaction (local memory access or configuration register access), it asserts PCI_DEVSEL one clock cycle following the address phase.

As an initiator, if the MPC8560 does not detect the assertion of PCI_DEVSEL within four clock cycles after the address phase (that is, five clock cycles after it asserts PCI_FRAME), it terminates the transaction with a master-abort termination; see Section 15.4.2.8.1, "Master-Initiated Termination."

15.4.2.5 Byte Alignment

The byte enable signals of the PCI bus (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 15.4.2.13.1, "PCI Parity," for more information.

If the MPC8560, 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 MPC8560 expects that the data is not changed, and on a write transaction, the data is not stored.

15.4.2.6 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 PCI_IRDY, PCI_TRDY, PCI_DEVSEL, and PCI_STOP signals use the address phase as their turnaround cycle. PCI_FRAME,PCI_C/BE[7:0], and PCI_AD[63:0] signals use the idle cycle between transactions (when both PCI_FRAME and PCI_IRDY are negated) as their turnaround cycle. The PCI_PERR signal has a turnaround cycle on the fourth clock after the last data phase.

The PCI address/data signals, 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 15.4.2.13.1, "PCI Parity," for more information.

15.4.2.7 PCI Bus Transactions

This section provides descriptions of the PCI bus transactions. All bus transactions follow the protocol as described in Section 15.4.2, "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.

15.4.2.7.1 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 PCI_FRAME. During the address phase, PCI_AD[63:0] contains a valid address and PCI_C/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 PCI_AD[63:0] as address signals to the target driving PCI_AD[63:0] as data signals. The turnaround cycle is enforced by the target with the TRDY signal. The target provides valid data at the earliest one cycle after the turnaround cycle. The target must drive the PCI_AD[63:0] signals when PCI_DEVSEL is asserted.

During the data phase, the PCI_C/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 PCI_C/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 PCI_IRDY and PCI_TRDY are asserted on the same clock edge. When either PCI_IRDY or PCI_TRDY is negated, a wait cycle is inserted and no data is transferred. The initiator indicates the last data phase by negating PCI_FRAME when PCI_IRDY is asserted. The transaction is considered complete when data is transferred in the last data phase.

Figure 15-47 illustrates a PCI single-beat read transaction. Figure 15-48 illustrates a PCI burst read transaction.

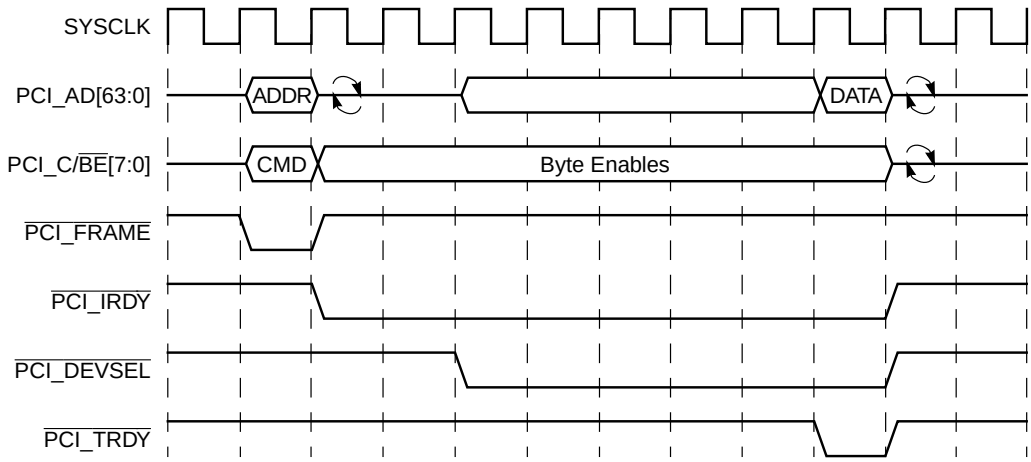


Figure 15-47. PCI Single-Beat Read Transaction

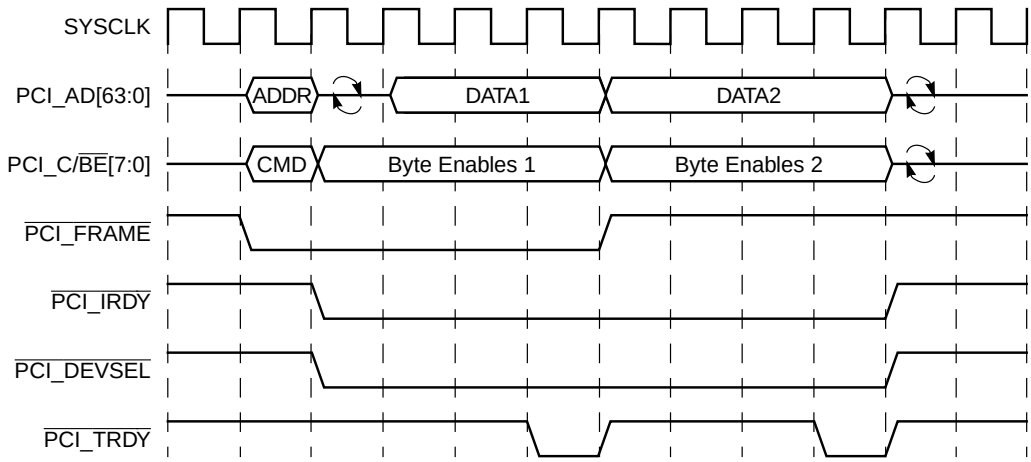


Figure 15-48. PCI Burst Read Transaction

15.4.2.7.2 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 15-49 illustrates a PCI single-beat write transaction. Figure 15-50 illustrates a PCI burst write transaction.

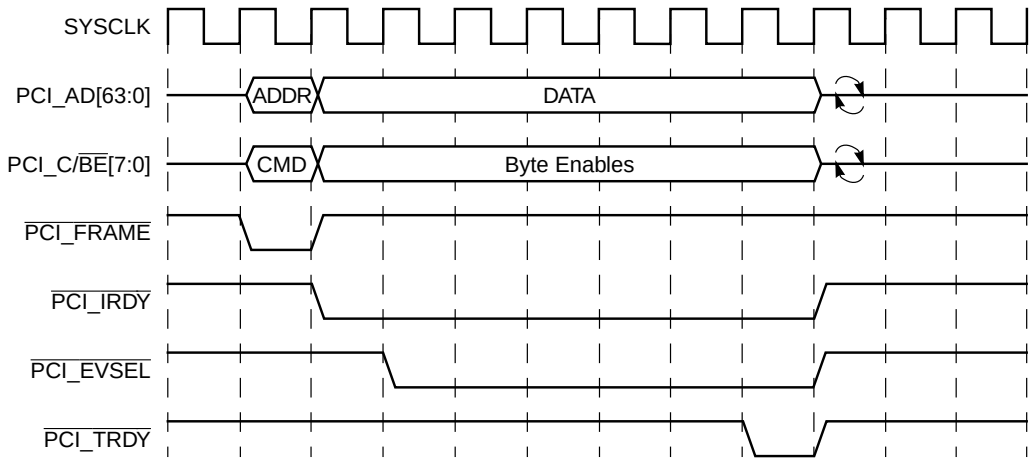


Figure 15-49. PCI Single-Beat Write Transaction

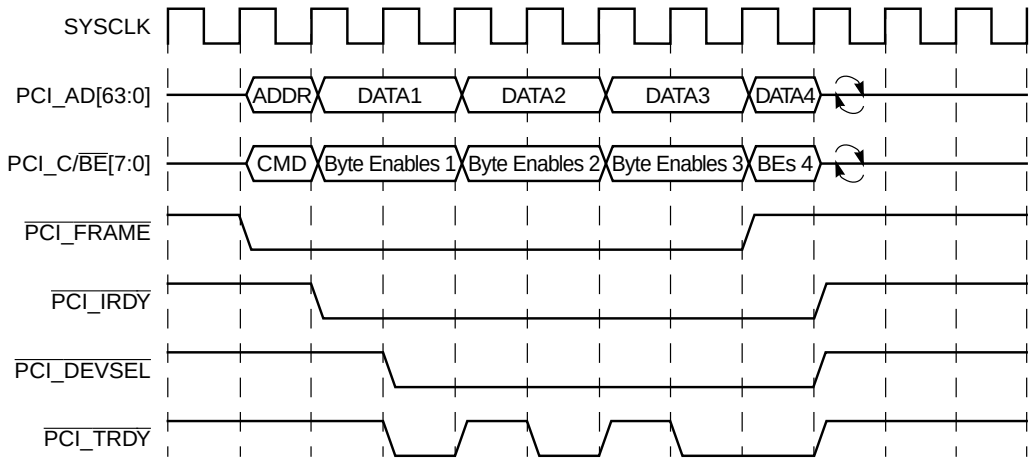


Figure 15-50. PCI Burst Write Transaction

15.4.2.8 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 PCI_FRAME and PCI_IRDY are both negated, indicating the bus is idle.

15.4.2.8.1 Master-Initiated Termination

Normally, a master initiates termination by negating PCI_FRAME and asserting PCI_IRDY. This indicates to the target that the final data phase is in progress. The final data transfer occurs when both PCI_TRDY and PCI_IRDY are asserted. The transaction is

considered complete when data is transferred in the last data phase. After the final data phase, both PCI_FRAME and 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 (GNT*n* 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 PCI_DEVSEL to claim a transaction, the initiator terminates the transaction with a master-abort. For a master-abort termination, the initiator negates PCI_FRAME and then negates 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 MPC8560 does not detect the assertion of PCI_DEVSEL within four clock cycles following the address phase (five clock cycles after asserting PCI_FRAME), it terminates the transaction with a master-abort. On reads that are master-aborted, the MPC8560 sends an error message back through the OCeaN to the internal requestor. On writes that are master-aborted, the data is discarded.

15.4.2.8.2 Target-Initiated Termination

By asserting the PCI_STOP signal, a target may request that the initiator terminate the current transaction. Once asserted, the target holds PCI_STOP asserted until the initiator negates PCI_FRAME. Data may or may not be transferred during the request for termination. If PCI_TRDY and PCI_IRDY are asserted during the assertion of PCI_STOP, data is transferred. However, if PCI_TRDY is negated when 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 PCI_STOP, the initiator must negate its REQ*n* signal for a minimum of two PCI clock cycles, (one corresponding to when the bus goes to the idle state (PCI_FRAME and PCI_IRDY negated)). If the initiator intends to complete the transaction, it can reassert its REQ*n* immediately following the two clock cycles. If the initiator does not intend to complete the transaction, it can assert REQ*n* 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

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 MPC8560 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 15.4.2.3.1, “Memory Space Addressing,” for more information.

As a target, the MPC8560 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 MPC8560 aborts the transaction internally, but continues the transaction on the PCI bus.

Figure 15-51 shows several target-initiated terminations.

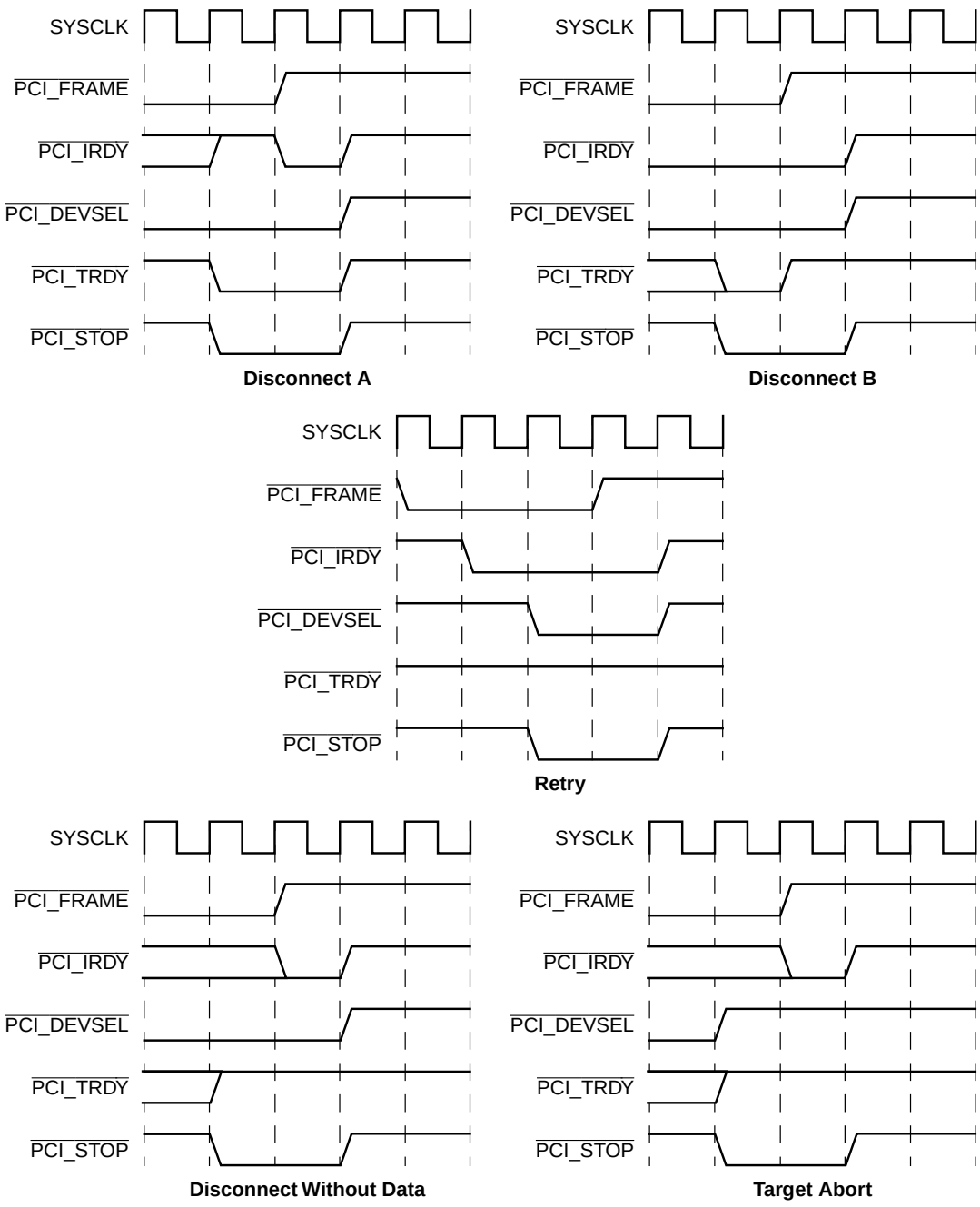


Figure 15-51. PCI Target-Initiated Terminations

The three disconnect terminations are unique in the data transferred at the end of the transaction. For disconnect A, the initiator is negating PCI_IRDY when the target asserts PCI_STOP and data is transferred only at the end of the current data phase. For disconnect B, the target negates PCI_TRDY one clock after it asserts 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 PCI_STOP when PCI_TRDY is negated indicating that the target cannot accept any more data.

15.4.2.9 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 PCI_FRAME is negated, and PCI_IRDY and 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 PCI_TRDY, PCI_DEVSEL, PCI_PERR, and 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 MPC8560 does not perform any fast back-to-back transactions. As a target, the MPC8560 supports both types of fast back-to-back transactions.

During fast back-to-back transactions, the MPC8560 monitors the bus states to determine if it is the target of a transaction. If the previous transaction was not directed to the MPC8560 and the current transaction is directed at the MPC8560, it delays the assertion of PCI_DEVSEL (as well as PCI_TRDY, PCI_STOP, and PCI_PERR) for one clock cycle to allow the other target to stop driving the bus.

15.4.2.10 Dual Address Cycles

The MPC8560 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 MPC8560 block supports single-beat and burst DAC transactions.

The timing sequence of the PCI signals for single-beat DAC reads is shown in Figure 15-52. The timing for a DAC burst read is shown in Figure 15-53. Figure 15-54 and Figure 15-55 show timing examples for single-beat DAC writes and burst DAC writes, respectively.

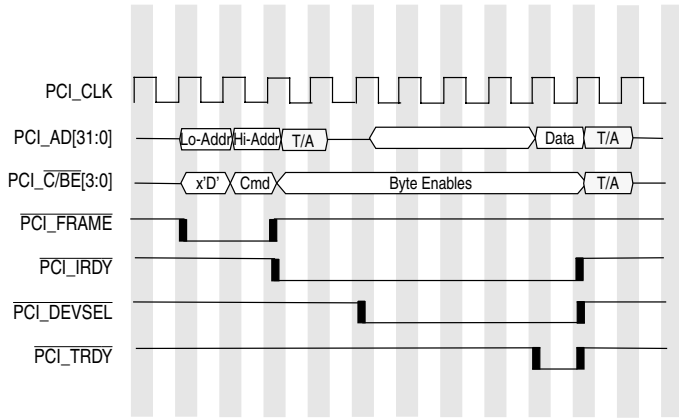


Figure 15-52. DAC Single-Beat Read Example

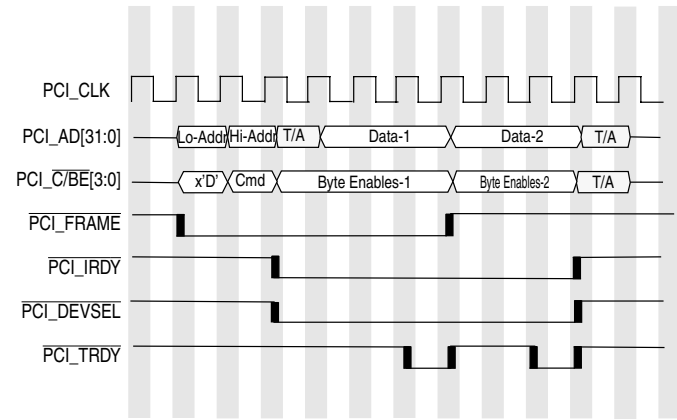


Figure 15-53. DAC Burst Read Example

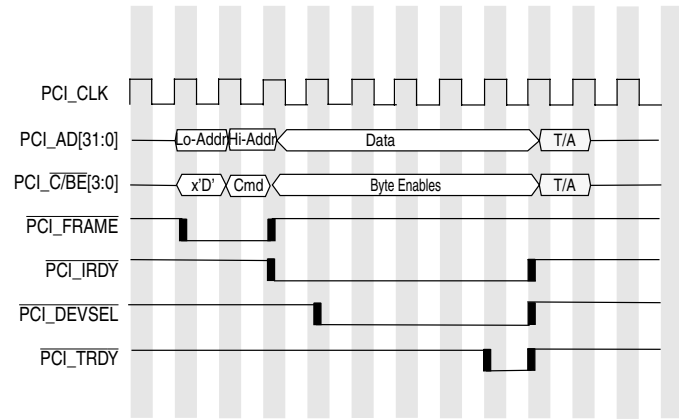


Figure 15-54. DAC Single-Beat Write Example

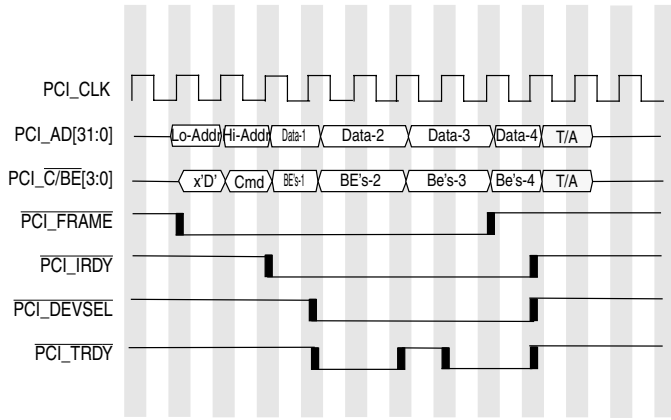


Figure 15-55. DAC Burst Write Example

15.4.2.11 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.

15.4.2.11.1 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 15-56. 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 15-56. The rest of the 256-byte configuration space is device-specific. The PCI header specific to the MPC8560 is described in Section 15.3.2, “PCI/PCI-X Configuration Header.”

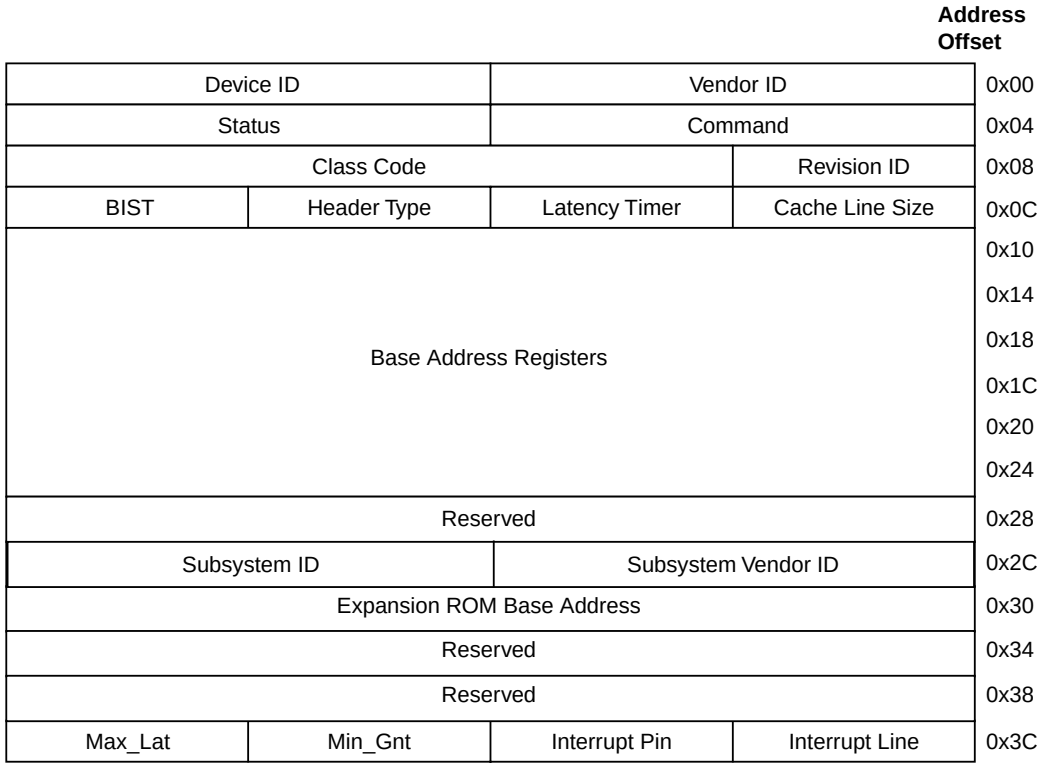


Figure 15-56. Standard PCI Configuration Header

Table 15-45 summarizes the configuration header registers. Detailed descriptions of these registers are provided in the *PCI Local Bus Specification*, Rev. 2.2.

Table 15-45. 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 15-45. 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 15-56 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 µs units
0x3F	Max_Lat	Specifies how often the device needs access to the bus in 0.25 µs units

15.4.2.11.2 Accessing the PCI Configuration Space

This section describes accessing the external PCI configuration space. See Section 15.3.2, "PCI/PCI-X Configuration Header," for information on accessing the internal configuration registers of the MPC8560.

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 MPC8560, 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 15-57.

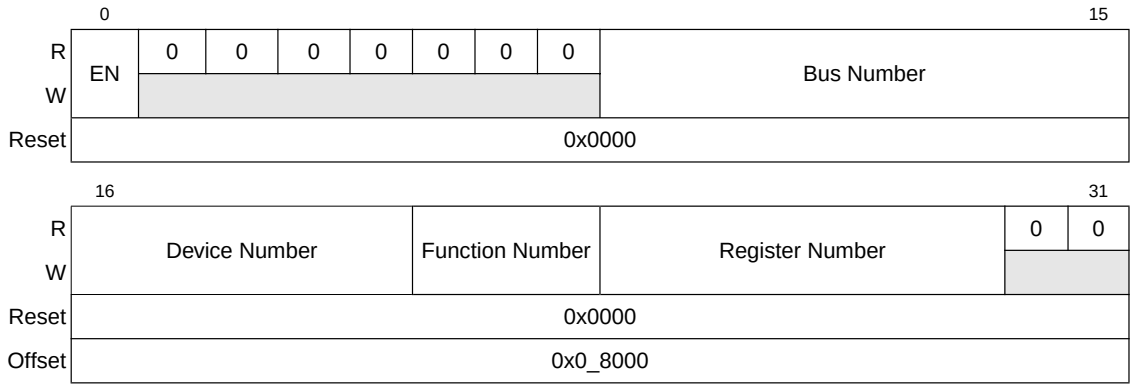


Figure 15-57. CFG_ADDR Register Format

Table 15-46 describes the fields within CFG_ADDR.

Table 15-46. 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 MPC8560, 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 15-58.

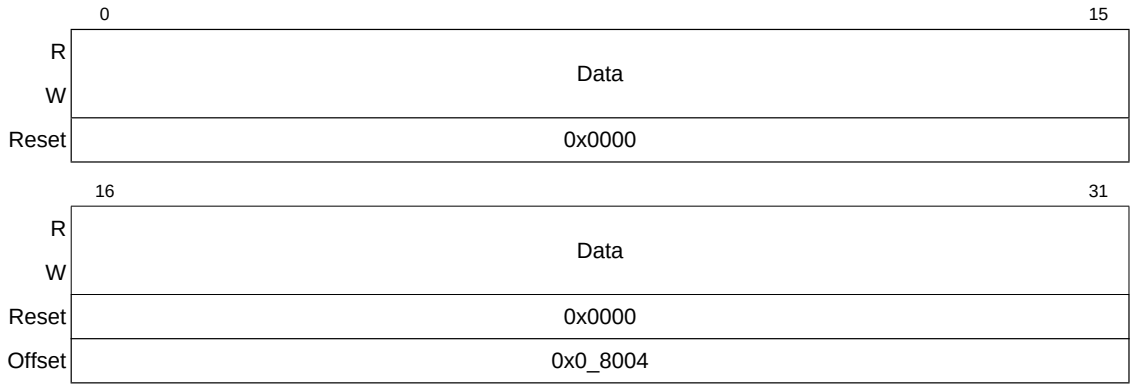


Figure 15-58. CFG_DATA Register

Note that CFG_DATA may contain 1, 2, 3, or 4 bytes depending on the size of the register being accessed.

When the MPC8560 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 MPC8560 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 15.4.2.12, “Other Bus Transactions,” for more information. If the bus number corresponds to the local PCI bus (bus number = 0x00), the MPC8560 performs a type 0 configuration cycle translation. If the bus number indicates a remote PCI bus (that is, nonlocal), the MPC8560 performs a type 1 configuration cycle translation.

15.4.2.11.3 Type 0 Configuration Translation

Figure 15-59 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.

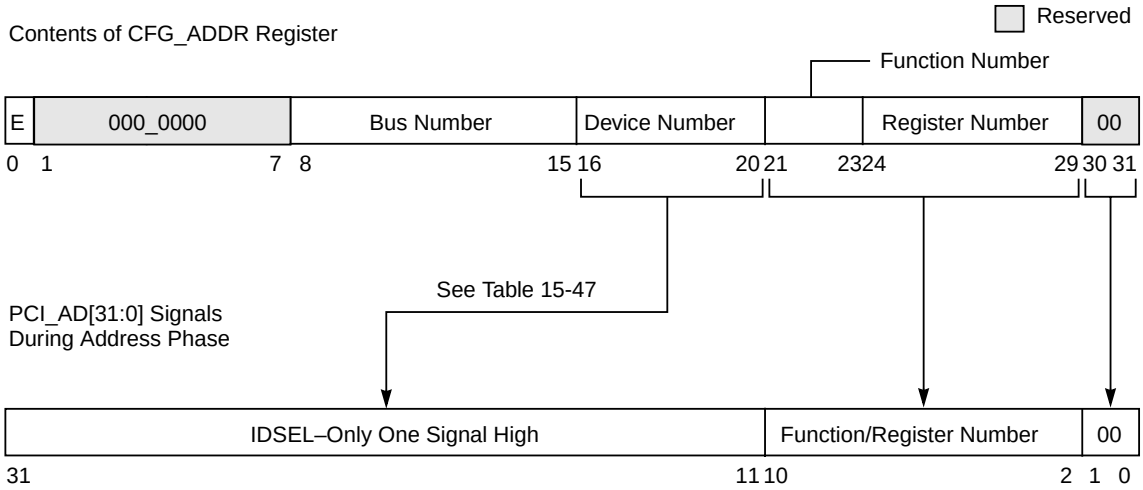


Figure 15-59. Type 0 Configuration Translation

For type 0 configuration cycles, the MPC8560 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 MPC8560 translates the device number to PCI_IDSEL as shown in Table 15-47.

Table 15-47. 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 15-47. 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	—

[1] A 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 MPC8560 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 $\overline{\text{PCI_FRAME}}$.

15.4.2.11.4 Type 1 Configuration Translation

For type 1 translations, the MPC8560 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 MPC8560 automatically translates PCI_AD[1:0] into 0b01 during the address phase to indicate a type 1 configuration cycle.

15.4.2.12 Other Bus Transactions

There are two other PCI transactions that the MPC8560 supports—interrupt acknowledge and special cycles. As an initiator, the MPC8560 may initiate both interrupt acknowledge and special-cycle transactions; however, as a target, the MPC8560 ignores interrupt-acknowledge and special-cycle transactions. Both transactions make use of the CFG_ADDR and CFG_DATA registers described in Section 15.4.2.11.2, "Accessing the PCI Configuration Space."

15.4.2.12.1 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 MPC8560 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 MPC8560 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 MPC8560 performs an interrupt-acknowledge transaction. If the bus number indicates a nonlocal PCI bus, the MPC8560 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 (PCI_C/$\overline{\text{BE}}$[3:0] = 0b0000). Although there is no explicit address, 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 MPC8560 PIC unit does not respond to PCI interrupt-acknowledge commands.

During the data phase, the responding device returns the interrupt vector on 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 PCI_C/$\overline{\text{BE}}$[7:0] signals.

The MPC8560 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.

15.4.2.12.2 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 MPC8560 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 MPC8560 performs a special-cycle transaction on the local PCI bus.

If the bus number indicates a nonlocal PCI bus, the MPC8560 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 (PCI_C/$\overline{\text{BE}}$[3:0] = 0b0001) the address phase contains no other valid information. Although there is no explicit address, PCI_AD[63:0] are driven to a stable state, and parity is generated. During the data phase, 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 (PCI_AD[15:0]); the optional data field is encoded on the most-significant 16 lines (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 15-48.

Table 15-48. Special-Cycle Message Encodings

PCI_AD[15:0]	Message
0x0000	SHUTDOWN
0x0001	HALT
0x0002	x86 Architecture-Specific
0x0003–0xFFFF	—

Note that the MPC8560 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.

15.4.2.13 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.

15.4.2.13.1 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 PCI_AD[31:0], 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/$\overline{BE}$[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/$\overline{BE}$[7:4] and PCI_C/$\overline{BE}$[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 15-60.

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.

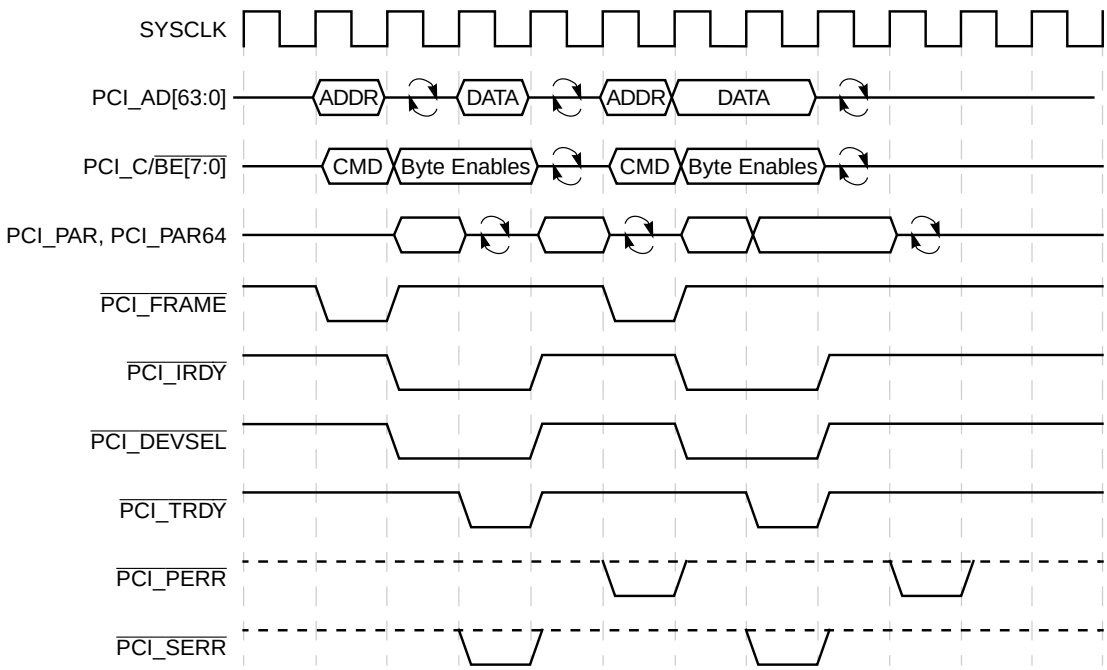


Figure 15-60. PCI Parity Operation

15.4.2.13.2 Error Reporting

PCI provides for the detection and signaling of both parity and other system errors. Two signals are used to report these errors—$\overline{\text{PCI_PERR}}$ and $\overline{\text{PCI_SERR}}$. The $\overline{\text{PCI_PERR}}$ signal is used exclusively to report data parity errors on all transactions except special cycles. The $\overline{\text{PCI_SERR}}$ 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.

15.5 Initialization/Application Information

15.5.1 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 15-49.

Table 15-49. 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 15-50.

Table 15-50. 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 15-50. 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

15.5.1.1 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.

15.5.1.2 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.

15.5.1.3 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.

15.5.2 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.

15.5.3 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 MPC8560. The MPC8560 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 16
RapidIO Interface

16.1 Introduction

The RapidIO interface conforms to Revision 1.2 of the Parallel *RapidIO Interconnect Specification*.

Figure 16-1 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 16.2, “External Signal Description,” describes the interface signal groups, their meanings, and timing relationships.
- Section 16.3, “Memory Map/Register Definition,” describes the configuration registers, runtime registers, and debug registers that constitute the programming model.
- Section 16.4.1, “RapidIO Transaction, Packet, and Control Symbol Summary” describes the functional organization and operation of the design.
- Section 16.5, “RapidIO Functionality”

16.1.1 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)

16.1.2 Block Diagram

Figure 16-1 is a block diagram of the RapidIO module.

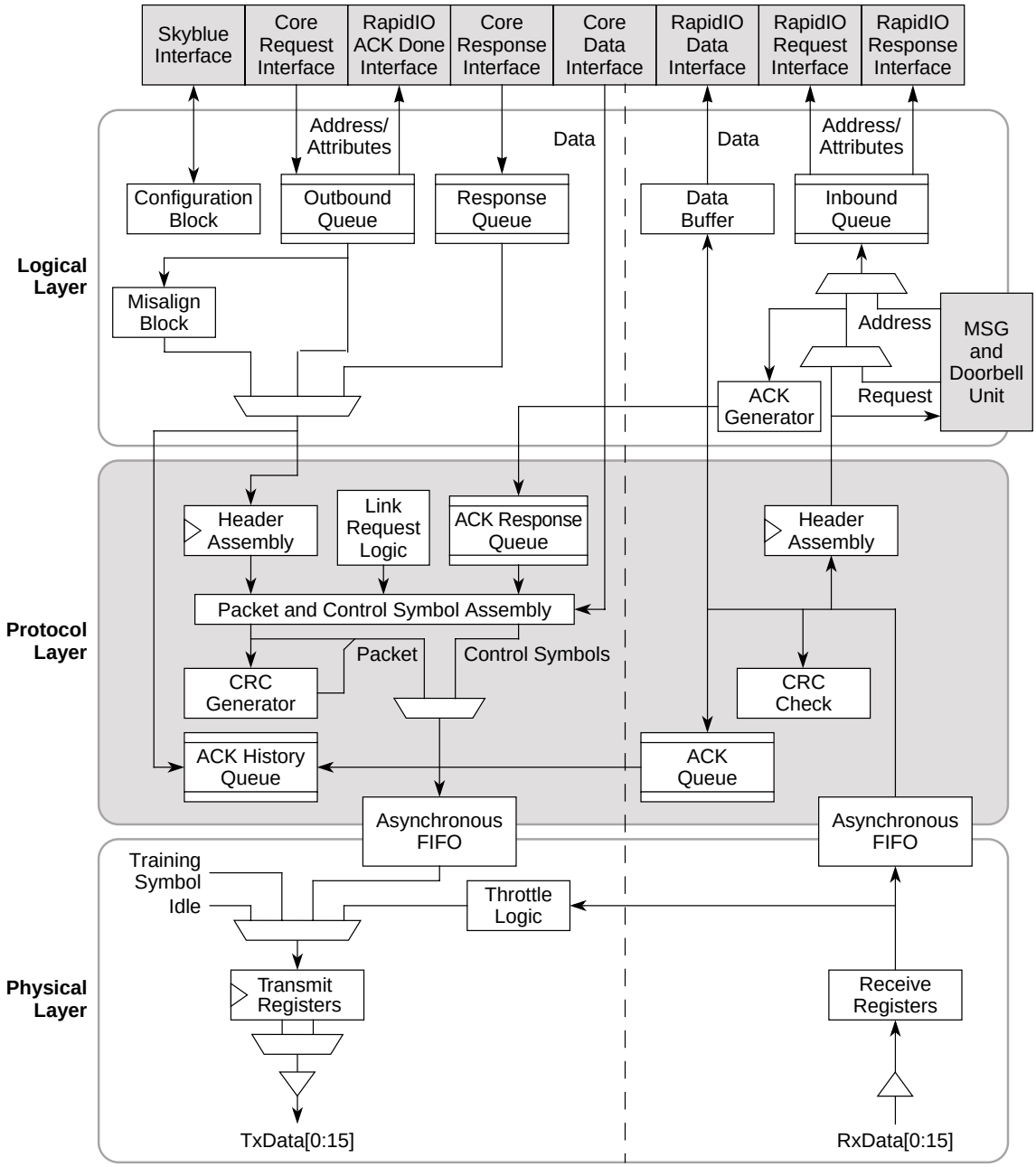


Figure 16-1. RapidIO High Level Block Diagram

16.1.3 RapidIO Interface Features

This section describes the RapidIO interface features.

16.1.3.1 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

16.1.3.2 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

16.1.4 RapidIO Modes of Operation

This section describes the RapidIO modes of operation.

16.1.4.1 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.

16.1.4.2 CRC Checking Modes

RapidIO offers following modes for packet CRC checking as shown in Table 16-1.

Table 16-1. CRC Checking Modes

Mode	crc_enable	crc_bypass
CRC enabled with hardware-based recovery	0	0
CRC checking disabled	1	Don't Care

16.1.4.3 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.

16.1.4.4 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.

16.1.4.5 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.

16.1.4.6 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 16-2. Accept All Mode Behavior

Accept All	Device Behavior for Target ID ! = Device ID
0	Error
1	Accept and process the packet without regard to the target ID

16.1.4.7 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 16.3.3.3.1, “Port Notification/Fatal Error Detect Register (PNFEDR).”

16.1.4.8 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 16.3.3.3.1, “Port Notification/Fatal Error Detect Register (PNFEDR).”

16.2 External Signal Description

This section describes the signal names of the RapidIO block, their meanings, and the timing relationships that exist among the signals.

16.2.1 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 16-2.

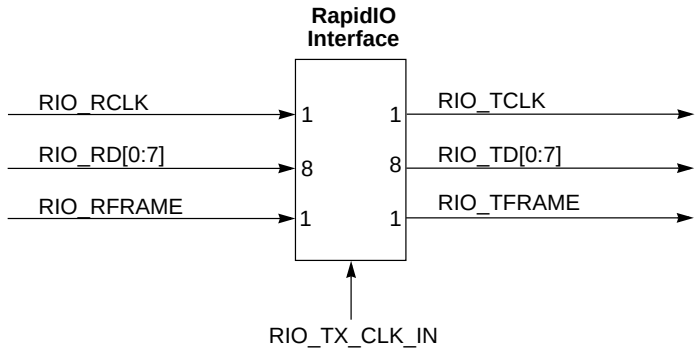


Figure 16-2. RapidIO Interface Signals

The RapidIO signals are implemented as differential pairs of signals. Table 16-3 lists the external RapidIO signals implemented on this interface.

Table 16-3. 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

16.2.2 Detailed Signal Descriptions

Table 16-4 describes the RapidIO interface signals in detail.

Table 16-4. 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]. $\overline{\text{RIO_RCLK}}$ 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 *MPC8560 Hardware Specification* 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. $\overline{\text{RIO_RD}}$[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. $\overline{\text{RIO_RFRAME}}$ 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). $\overline{\text{RIO_TCLK}}$ 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 16-4. 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. $\overline{\text{RIO_TD}}$[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. $\overline{\text{RIO_TFRAME}}$ _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. $\overline{\text{RIO_TX_CLK_IN}}$ is the differential pair of the RIO_TX_CLK_IN signal.	
		State Meaning	Asserted/Negated—
		Timing	Assertion/Negation—

16.3 Memory Map/Register Definition

16.3.1 RapidIO Registers

The RapidIO memory map is shown in Table 16-5.

Table 16-5. RapidIO Module Memory Map

Offset	Register	Access	Reset	Section/Page
Architectural Registers				
0xC_0000	Device identity capability register (DIDCAR)	R	0x0003_0002	16.3.2.1/16-14
0xC_0004	Device information capability register (DICAR)	R	0x8060_0010	16.3.2.2/16-14
0xC_0008	Assembly identity capability register (AIDCAR)	Special [1]	0x0000_0000	16.3.2.3/16-15
0xC_000C	Assembly information capability register (AICAR)	Special[1]	0x0000_0100	16.3.2.4/16-16
0xC_0010	Processing element features capability register (PEFCAR)	R	0xE088_0009	16.3.2.5/16-16
0xC_0014	Switch port information capability register (SPICAR)	R	0x0000_0100	16.3.2.6/16-17
0xC_0018	Source operations capability register (SOCAR)	R	0x0600_FCF0	16.3.2.7/16-18
0xC_001C	Destination operations capability register (DOCAR)	R	0x0600_FCF4	16.3.2.8/16-19
0xC_0040	Mailbox command and status register (MSR)	R	0x0000_0000	16.3.2.9/16-21

Table 16-5. 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	16.3.2.10/16-22
0xC_004C	Processing Element Logical Layer Control Command and Status Register (PELLCCSR)	R	0x0000_0001	16.3.2.11/16-23
0xC_0058	Reserved	R	0x0000_0000	—
0xC_005C	Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)	R/W	0x0000_0000	16.3.2.12/16-23
0xC_0060	Base Device ID Command and Status Register (BDIDCSR)	R/W	0x00[cfg]_0000	16.3.2.13/16-24
0xC_0068	Host Base Device ID Lock Command and Status Register (HBDIDLCSR)	Special[1]	0x0000_FFFF	16.3.2.14/16-25
0xC_006C	Component Tag Command and Status Register (CTCSR)	R/W	0x0000_0000	16.3.2.15/16-25
0xC_0100	8/16 LP-LVDS Port Maintenance Block Header 0 Command and Status Register (PMBH0CSR)	R	0x0000_0002	16.3.2.16/16-26
0xC_0120	Port Link Time-Out Control Command and Status Register (PLTOCCSR)	R/W	0xFFFF_FF00	16.3.2.17/16-27
0xC_0124	Port Response Time-Out Control Command and Status Register (PRTOCCSR)	R/W	0xFFFF_FF00	16.3.2.18/16-27
0xC_013C	Port General Control Command and Status Register (PGCCSR)	R/W	0x[cfg]000_0000	16.3.2.19/16-28
0xC_0140	Port Link Maintenance Request Command and Status Register (PLMREQCSR)	R/W	0x0000_0000	16.3.2.19/16-28
0xC_0144	Port Link Maintenance Response Command and Status Register (PLMRESPCSR)	R	0x0000_0000	16.3.2.20/16-29
0xC_0148	Port Local Ackid Status Command and Status Register (PLASCSR)	Special[1]	0x0000_0000	16.3.2.21/16-29
0xC_0158	Port Error and Status Command and Status Register (PESCSR)	Special[1]	0x0000_0000	16.3.2.22/16-30
0xC_015C	Port Control Command and Status Register (PCCSR)	Special[1]	0x4400_0000	16.3.2.23/16-31
Implementation Registers				
0xD_0000	Configuration Register (CR)	R/W	0x0000_0001	16.3.3.1.1/16-33
0xD_0010	Port Configuration Register (PCR)	R/W	0x0000_0010	16.3.3.1.2/16-34
0xD_0014	Port Error Injection Register (PEIR)	Special[1]	0x0000_0000	16.3.3.1.3/16-35
ATMU Registers				
0xD_0C00	RapidIO Outbound Window Translation Address Register 0 (ROWTAR0)	R/W	0x0000_0000	16.3.3.2.1/16-36
0xD_0C10	RapidIO Outbound Window Attributes Register 0 (ROWAR0)	R/W	0x0004_401F	16.3.3.2.3/16-39

Table 16-5. 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	16.3.3.2.1/16-36
0xD_0C28	RapidIO Outbound Window Base Address Register 1 (ROWBAR1)	R/W	0x0000_0000	16.3.3.2.2/16-38
0xD_0C30	RapidIO Outbound Window Attributes Register 1 (ROWAR1)	R/W	0x0004_401F	16.3.3.2.3/16-39
0xD_0C40	RapidIO Outbound Window Translation Address Register 2 (ROWTAR2)	R/W	0x0000_0000	16.3.3.2.1/16-36
0xD_0C48	RapidIO Outbound Window Base Address Register 2 (ROWBAR2)	R/W	0x0000_0000	16.3.3.2.2/16-38
0xD_0C50	RapidIO Outbound Window Attributes Register 2 (ROWAR2)	R/W	0x0004_401F	16.3.3.2.3/16-39
0xD_0C60	RapidIO Outbound Window Translation Address Register 3 (ROWTAR3)	R/W	0x0000_0000	16.3.3.2.1/16-36
0xD_0C68	RapidIO Outbound Window Base Address Register 3 (ROWBAR3)	R/W	0x0000_0000	16.3.3.2.2/16-38
0xD_0C70	RapidIO Outbound Window Attributes Register 3 (ROWAR3)	R/W	0x0004_401F	16.3.3.2.3/16-39
0xD_0C80	RapidIO Outbound Window Translation Address Register 4 (ROWTAR4)	R/W	0x0000_0000	16.3.3.2.1/16-36
0xD_0C88	RapidIO Outbound Window Base Address Register 4 (ROWBAR4)	R/W	0x0000_0000	16.3.3.2.2/16-38
0xD_0C90	RapidIO Outbound Window Attributes Register 4 (ROWAR4)	R/W	0x0004_401F	16.3.3.2.3/16-39
0xD_0CA0	RapidIO Outbound Window Translation Address Register 5 (ROWTAR5)	R/W	0x0000_0000	16.3.3.2.1/16-36
0xD_0CA8	RapidIO Outbound Window Base Address Register 5 (ROWBAR5)	R/W	0x0000_0000	16.3.3.2.2/16-38
0xD_0CB0	RapidIO Outbound Window Attributes Register 5 (ROWAR5)	R/W	0x0004_401F	16.3.3.2.3/16-39
0xD_0CC0	RapidIO Outbound Window Translation Address Register 6 (ROWTAR6)	R/W	0x0000_0000	16.3.3.2.1/16-36
0xD_0CC8	RapidIO Outbound Window Base Address Register 6 (ROWBAR6)	R/W	0x0000_0000	16.3.3.2.2/16-38
0xD_0CD0	RapidIO Outbound Window Attributes Register 6 (ROWAR6)	R/W	0x0004_401F	16.3.3.2.3/16-39
0xD_0CE0	RapidIO Outbound Window Translation Address Register 7 (ROWTAR7)	R/W	0x0000_0000	16.3.3.2.1/16-36
0xD_0CE8	RapidIO Outbound Window Base Address Register 7 (ROWBAR7)	R/W	0x0000_0000	16.3.3.2.2/16-38

Table 16-5. RapidIO Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_0CF0	RapidIO Outbound Window Attributes Register 7 (ROWAR7)	R/W	0x0004_401F	16.3.3.2.3/16-39
0xD_0D00	RapidIO Outbound Window Translation Address Register 8 (ROWTAR8)	R/W	0x0000_0000	16.3.3.2.1/16-36
0xD_0D08	RapidIO Outbound Window Base Address Register 8 (ROWBAR8)	R/W	0x0000_0000	16.3.3.2.2/16-38
0xD_0D10	RapidIO Outbound Window Attributes Register 8 (ROWAR8)	R/W	0x0004_401F	16.3.3.2.3/16-39
0xD_0D60	RapidIO Inbound Window Translation Address Register 4 (RIWTAR4)	R/W	0x0000_0000	16.3.3.2.4/16-41
0xD_0D68	RapidIO Inbound Window Base Address Register 4 (RIWBAR4)	R/W	0x0000_0000	16.3.3.2.5/16-42
0xD_0D70	RapidIO Inbound Window Attributes Register 4 (RIWAR4)	R/W	0x0004_401F	16.3.3.2.6/16-42
0xD_0D80	RapidIO Inbound Window Translation Address Register 3 (RIWTAR3)	R/W	0x0000_0000	16.3.3.2.4/16-41
0xD_0D88	RapidIO Inbound Window Base Address Register 3 (RIWBAR3)	R/W	0x0000_0000	16.3.3.2.5/16-42
0xD_0D90	RapidIO Inbound Window Attributes Register 3 (RIWAR3)	R/W	0x0004_401F	16.3.3.2.6/16-42
0xD_0DA0	RapidIO Inbound Window Translation Address Register 2 (RIWTAR2)	R/W	0x0000_0000	16.3.3.2.4/16-41
0xD_0DA8	RapidIO Inbound Window Base Address Register 2 (RIWBAR2)	R/W	0x0000_0000	16.3.3.2.5/16-42
0xD_0DB0	RapidIO Inbound Window Attributes Register 2 (RIWAR2)	R/W	0x0004_401F	16.3.3.2.6/16-42
0xD_0DC0	RapidIO Inbound Window Translation Address Register 1 (RIWTAR1)	R/W	0x0000_0000	16.3.3.2.4/16-41
0xD_0DC8	RapidIO Inbound Window Base Address Register 1 (RIWBAR1)	R/W	0x0000_0000	16.3.3.2.5/16-42
0xD_0DD0	RapidIO Inbound Window Attributes Register1 (RIWAR1)	R/W	0x0004_401F	16.3.3.2.6/16-42
0xD_0DE0	RapidIO Inbound Window Translation Address Register 0 (RIWTAR0)	R/W	0x0000_0000	16.3.3.2.4/16-41
0xD_0DF0	RapidIO Inbound Window Attributes Register 0 (RIWAR0)	R/W	0x0004_401F	16.3.3.2.6/16-42
Error Management Registers				
0xD_0E00	Port Notification/Fatal Error Detect Register (PNFEDR)	Special[1]	0x0000_0000	16.3.3.3.1/16-44
0xD_0E04	Port Notification/Fatal Error Disable Register (PNFEDiR)	R/W	0x0000_0000	16.3.3.3.2/16-47

Table 16-5. 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	16.3.3.3.3/16-49
0xD_0E0C	Port Error Control Register (PECR)	Special[1]	0x0000_0000	16.3.3.3.4/16-50
0xD_0E10	Port Error Packet/Control Symbol Register 0 (PEPCSR0)	Special[1]	0x0000_0000	16.3.3.3.5/16-51
0xD_0E14	Port Error Packet Register 1 (PEPR1)	Special[1]	0x0000_0000	16.3.3.3.6/16-52
0xD_0E18	Port Error Packet Register 2 (PEPR2)	Special[1]	0x0000_0000	16.3.3.3.16/16-58
0xD_0E20	Port Recoverable Error Detect Register (PREDR)	Special[1]	0x0000_0000	16.3.3.3.23/16-62
0xD_0E28	Port Error Recovery Threshold Register (PERTR)	Special[1]	0x00FF_0000	16.3.3.3.24/16-65
0xD_0E2C	Port Retry Threshold Register (PRTR)	Special[1]	0x00FF_0000	16.3.3.3.25/16-66
RapidIO Message Unit				
RapidIO Outbound Message Registers				
0xD_1000	Outbound Mode Register (OMR)	R/W	0x0000_0000	16.3.4.1.1/16-67
0xD_1004	Outbound Status Register (OSR)	Special	0x0000_0000	16.3.4.1.2/16-69
0xD_100C	Outbound Descriptor Queue Dequeue PointerAddress Register (ODQDPAR)	R/W	0x0000_0000	16.3.4.1.3/16-70
0xD_1014	Outbound Source Address Register (OSAR)	R/W	0x0000_0000	16.3.4.1.4/16-71
0xD_1018	Outbound Destination Port Register (ODPR)	R/W	0x0000_0000	16.3.4.1.5/16-72
0xD_101C	Outbound Destination Attributes Register (ODATR)	R/W	0x0006_0000	16.3.4.1.6/16-72
0xD_1020	Outbound Double-word Count Register (ODCR)	R/W	0x0000_0000	16.3.4.1.7/16-73
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	16.3.4.2.1/16-75
0xD_1064	Inbound Mailbox Status Register (ISR)	Special	0x0000_0000	16.3.4.2.2/16-77
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	16.3.4.2.4/16-79
RapidIO Doorbell Registers				
0xD_1460	Doorbell Mode Register (DMR)	R/W	0x0000_0000	16.3.4.3.1/16-80
0xD_1464	Doorbell Status Register (DSR)	Special	0x0000_0000	16.3.4.3.2/16-82
0xD_146C	Doorbell Queue Dequeue Pointer Address Register (DQDPAR)	R/W	0x0000_0000	16.3.4.3.3/16-83
0xD_1474	Doorbell Queue Enqueue Pointer Address Register (DQEPAR)	R/W	0x0000_0000	17.3.4.3.4/17-99

Table 16-5. 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	16.3.4.4.1/16-85
0xD_14E4	Port-write Status Register (PWSR)	Special	0x0000_0000	16.3.4.4.2/16-86
0xD_14EC	Port-write Queue Base Address Register (PWQBAR)	R/W	0x0000_0000	16.3.4.4.3/16-87

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

16.3.2 Architectural Registers

Following are the architectural registers of the RapidIO module.

16.3.2.1 Device Identity Capability Register (DIDCAR)

The DIDCAR, shown in Figure 16-3, is a read only register.

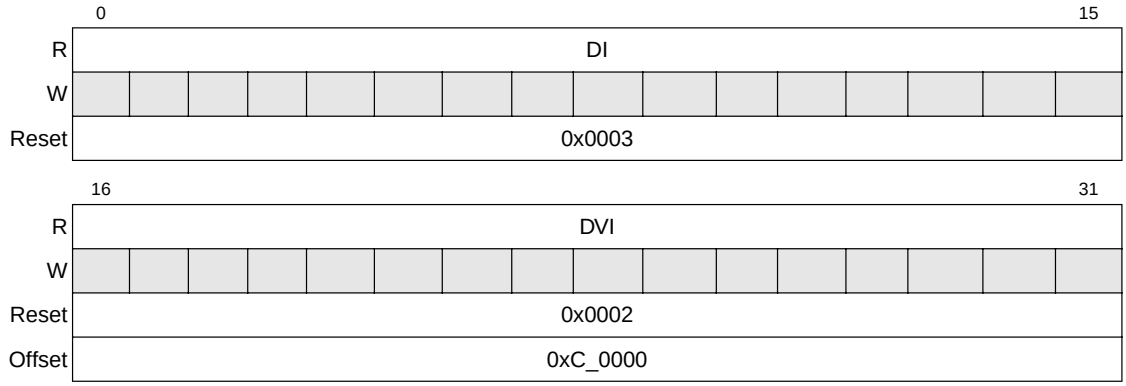


Figure 16-3. Device Identity Capability Register (DIDCAR)

Table 16-6 describes the bits of DIDCAR.

Table 16-6. DIDCAR Field Descriptions

Bits	Name	Description
0–15	DI	Device identity field ... 0x0003—MPC8560
16–31	DVI	Device vendor identity field 0x0002—Motorola

16.3.2.2 Device Information Capability Register (DICAR)

The DICAR, shown in Figure 16-4, 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.

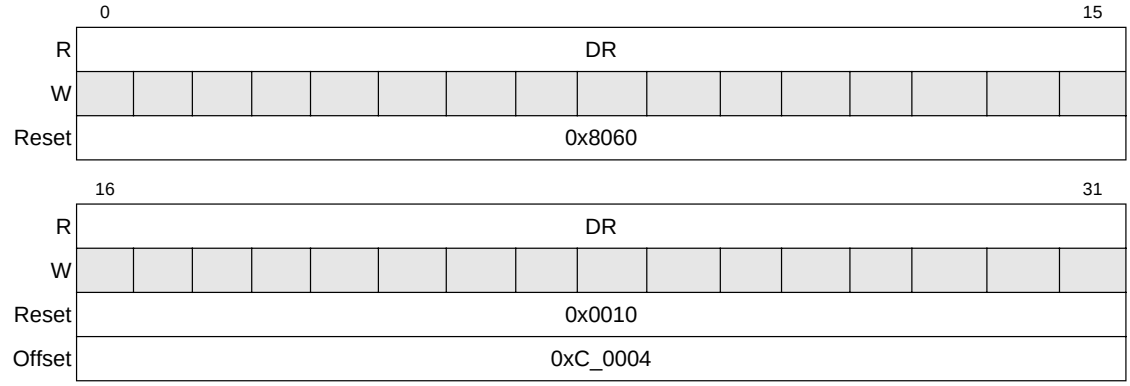


Figure 16-4. Device Information Capability Register (DICAR)

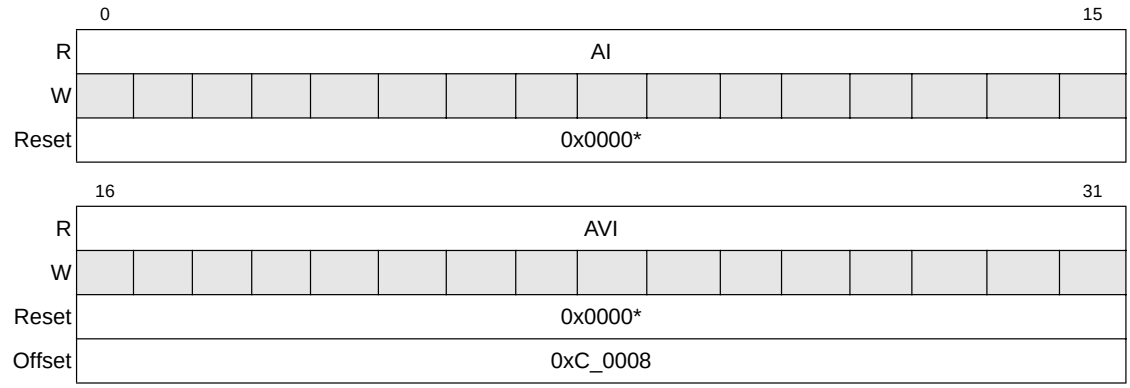
Table 16-7 describes the bits of DICAR.

Table 16-7. DICAR Field Descriptions

Bits	Name	Description
0–31	DR	Device revision

16.3.2.3 Assembly Identity Capability Register (AIDCAR)

The AIDCAR, shown in Figure 16-5, is a read only register.



* Alternate value may be read from serial ROM

Figure 16-5. Assembly Identity Capability Register (AIDCAR)

Table 16-8 describes the bits of AIDCAR.

Table 16-8. AIDCAR Field Descriptions

Bits	Name	Description
0–15	AI	Assembly identity field
16–31	AVI	Assembly vendor identity field

16.3.2.4 Assembly Information Capability Register (AICAR)

The AICAR, shown in Figure 16-6, is a read only register.

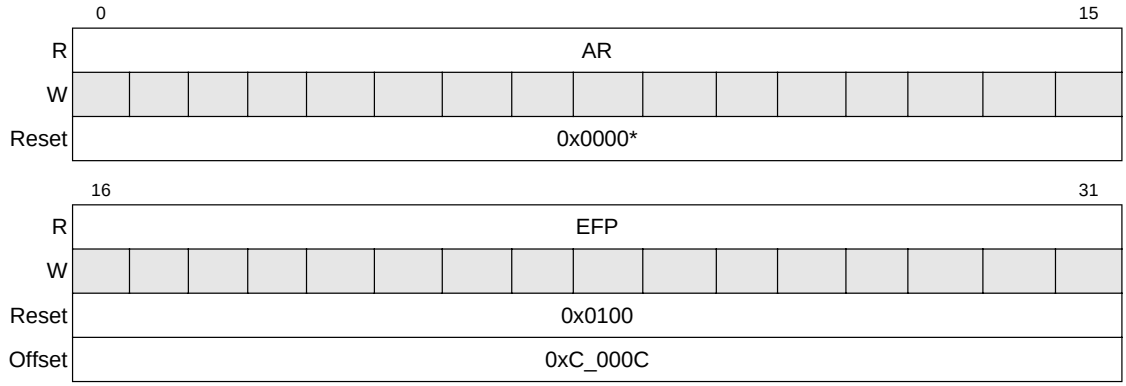


Figure 16-6. Assembly Information Capability Register (AICAR)

Table 16-9 describes the bits of AICAR.

Table 16-9. AICAR Field Descriptions

Bits	Name	Description
0–15	AR	Assembly revision field
16–31	EFP	Extended features pointer field

16.3.2.5 Processing Element Features Capability Register (PEFCAR)

The PEFCAR, shown in Figure 16-7, 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 16-7. Processing Element Features Capability Register (PEFCAR)

Table 16-10 describes the bits of PEFCAR.

Table 16-10. 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

16.3.2.6 Switch Port Information Capability Register (SPICAR)

The SPICAR, shown in Figure 16-8, is a read-only register.

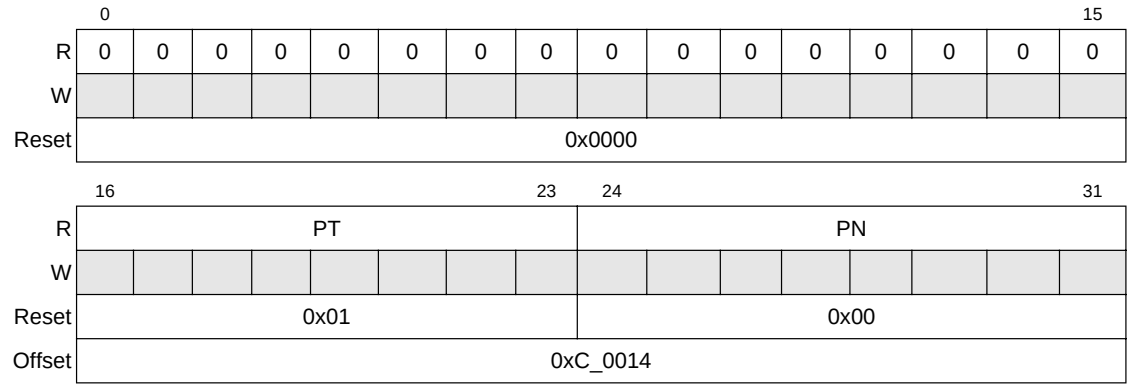


Figure 16-8. Switch Port Information Capability Register (SPICAR)

Table 16-11 describes the bits of SPICAR.

Table 16-11. 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

16.3.2.7 Source Operations Capability Register (SOCAR)

The SOCAR, shown in Figure 16-9, is a read only register. It indicates the RapidIO transaction types that this device can initiate.

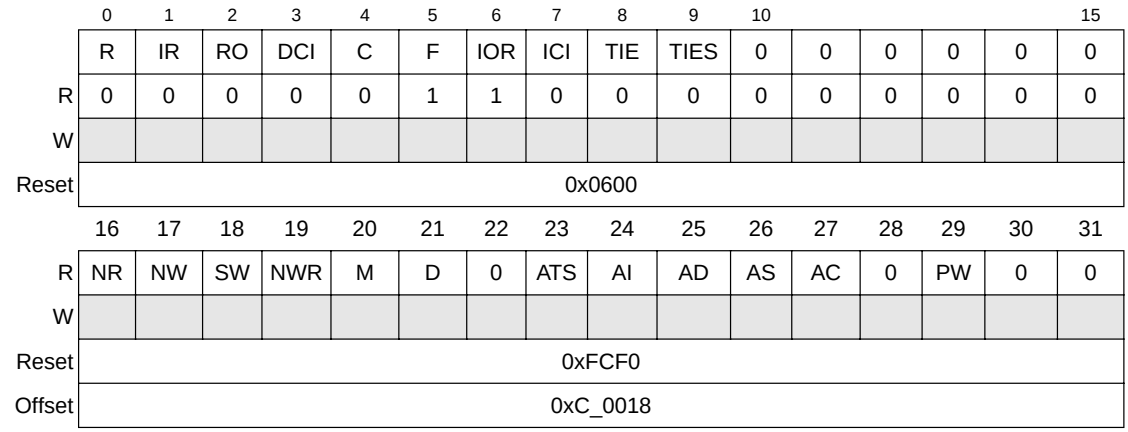


Figure 16-9. Source Operations Capability Register (SOCAR)

Table 16-12 describes the bits of SOCAR.

Table 16-12. 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

16.3.2.8 Destination Operations Capability Register (DOCAR)

The DOCAR, shown in Figure 16-10, is a read only register. It indicates the RapidIO transaction types that this device can service.

	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 16-10. Destination Operations Capability Register (DOCAR)

Table 16-13 describes the bits of DOCAR.

Table 16-13. 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 16-13. 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

16.3.2.9 Mailbox Command and Status Register (MSR)

The MSR, shown in Figure 16-11, is a read-only register.

	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															31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0x0000															
Offset	0xC_0040															

Figure 16-11. Mailbox Status Register (MSR)

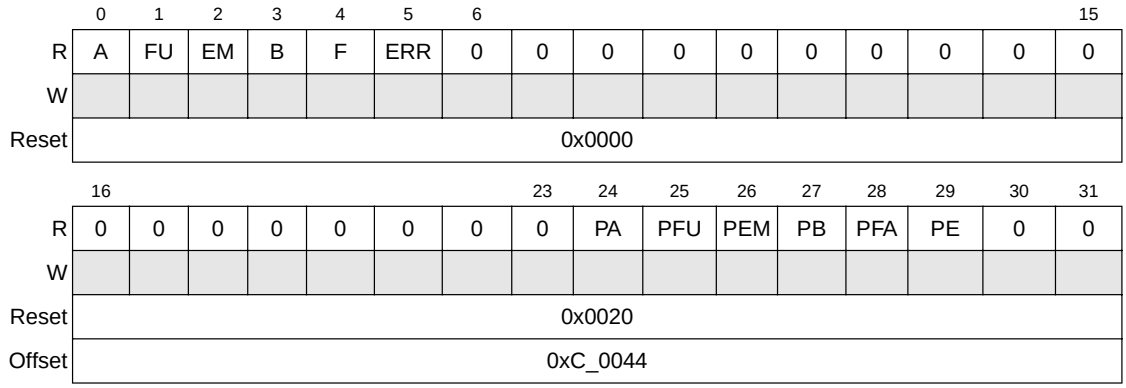
Table 16-14 describes the bits of MSR.

Table 16-14. 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

16.3.2.10 Port-Write and Doorbell Command and Status Register (PWDCSR)

The PWDCSR, shown in Figure 16-12, 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 16-12. Port-Write and Doorbell Command and Status Register (PWDCSR)

Table 16-15 describes the bits of PWDCSR.

Table 16-15. 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

16.3.2.11 Processing Element Logic Layer Control Command and Status Register (PELLCCSR)

The PELLCCSR, shown in Figure 16-13, is a read only register. Details of this register can be found in the RapidIO 1.2 Specification.

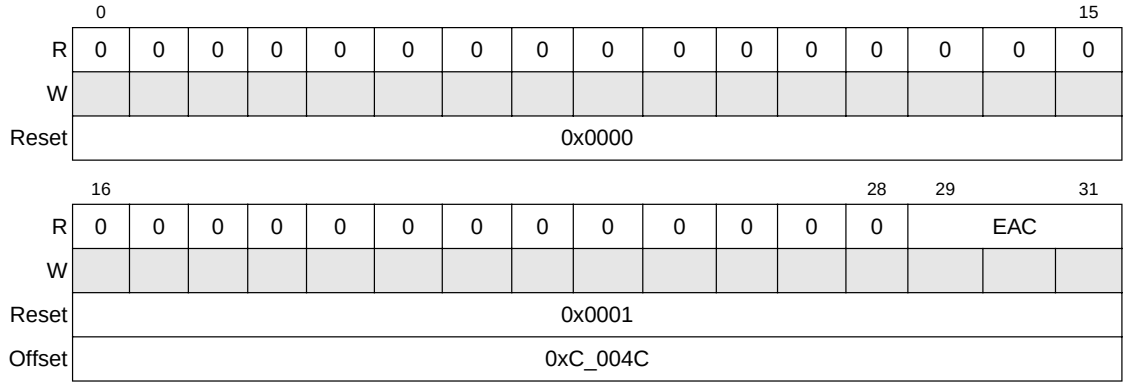


Figure 16-13. Processing Element Logic Layer Control Command and Status Register (PELLCCSR)

Table 16-16 describes the bits of PELLCCSR.

Table 16-16. PELLCCSR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29–31	EAC	Extended addressing control. The RapidIO controller supports 34-bit addresses. (Read only)

16.3.2.12 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 16-14, 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.

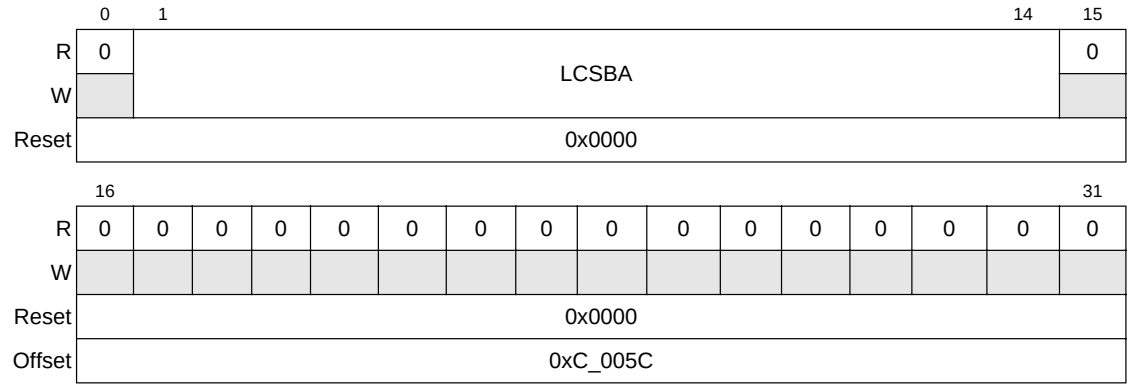


Figure 16-14. Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)

Table 16-17 describes the bits of LCSBA1CSR.

Table 16-17. 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

16.3.2.13 Base Device ID Command and Status Register (BDIDCSR)

The BDIDCSR is shown in Figure 16-15.

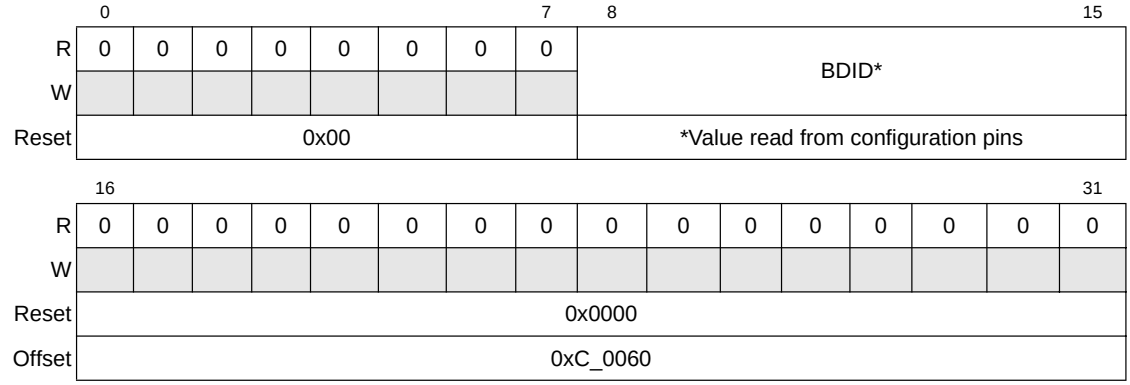


Figure 16-15. Base Device ID Command and Status Register (BDIDCSR)

BDIDCSR fields are described in Table 16-18.

Table 16-18. 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

16.3.2.14 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 16-16.

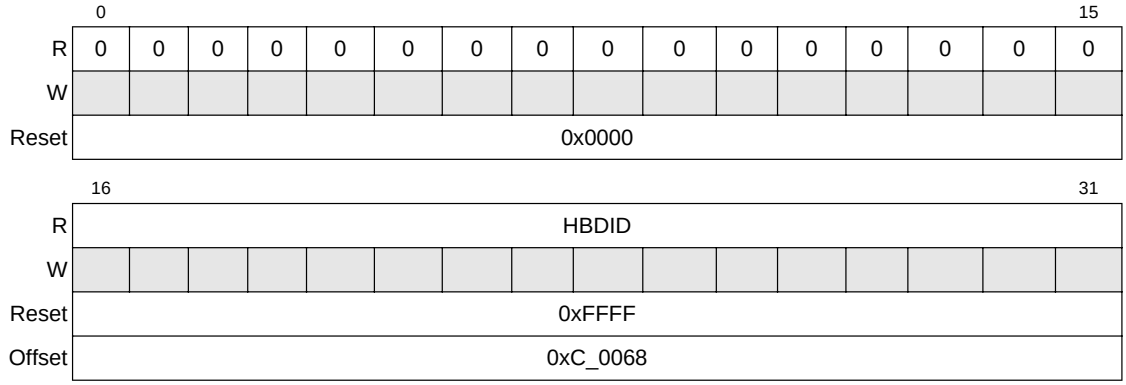


Figure 16-16. Host Base Device ID Lock Command and Status Register (HBDIDLCSR)

HBDIDLCSR fields are described in Table 16-19.

Table 16-19. 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.

16.3.2.15 Component Tag Command and Status Register (CTCSR)

The CTCSR is shown in Figure 16-17.

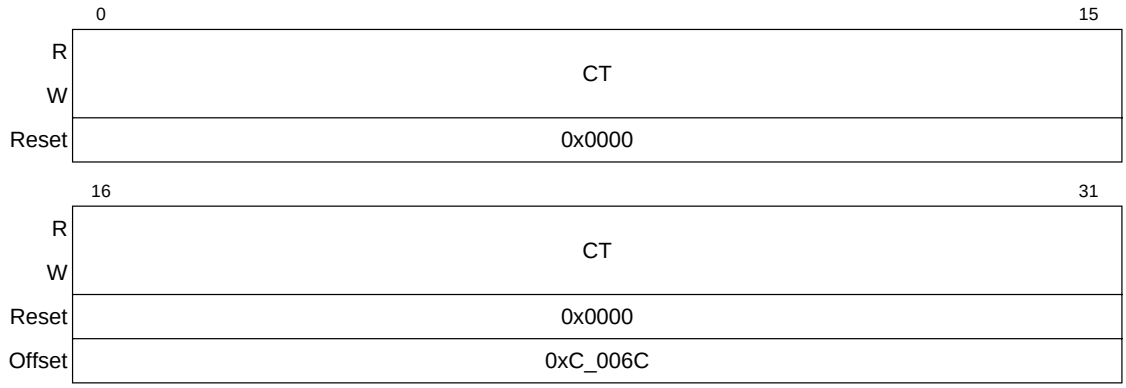


Figure 16-17. Component Tag Command and Status Register (CTCSR)

CTCSR fields are described in Table 16-20.

Table 16-20. CTCSR Field Descriptions

Bits	Name	Description
0–31	CT	Component tag

16.3.2.16 8/16 LP-LVDS Port Maintenance Block Header 0 Command and Status Register (PMBH0CSR)

The PMBH0CSR is shown in Figure 16-18.

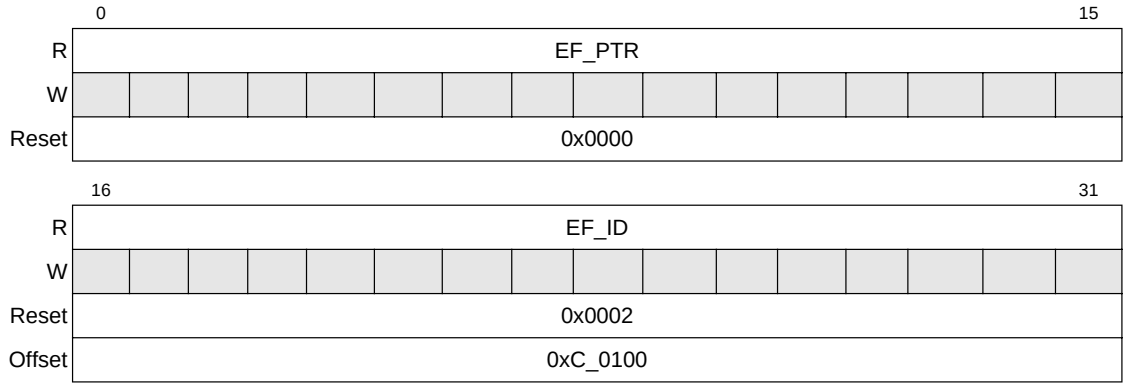


Figure 16-18. 8/16 LP-LVDS Port Maintenance Block Header 0 Command and Status Register (PMBH0CSR)

PMBH0CSR fields are described in Table 16-21.

Table 16-21. PMBH0CSR Field Descriptions

Bits	Name	Description
0–15	EF_PTR	Extended features pointer
16–31	EF_ID	Extended features ID

16.3.2.17 Port Link Time-Out Control Command and Status Register (PLTOCCSR)

The PLTOCCSR is shown in Figure 16-19.

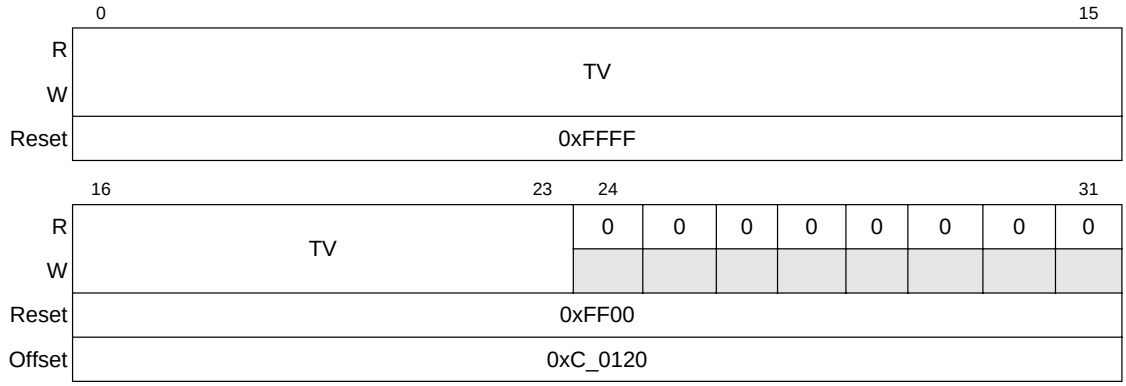


Figure 16-19. Port Link Time-Out Control Command and Status Register (PLTOCCSR)

PLTOCCSR fields are described in Table 16-22.

Table 16-22. PLTOCCSR Field Descriptions

Bits	Name	Description
0–23	TV	Time-out value
24–31	—	Reserved

16.3.2.18 Port Response Time-Out Control Command and Status Register (PRTOCCSR)

The PRTOCCSR is shown in Figure 16-20.

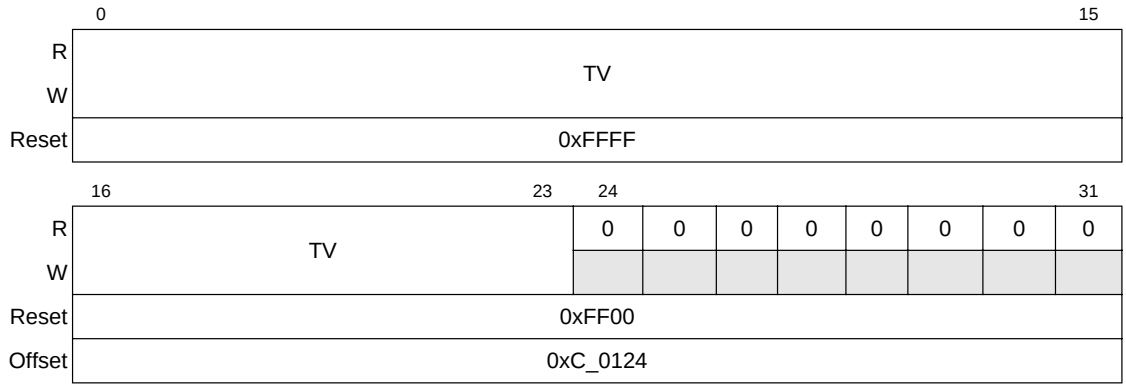


Figure 16-20. Port Response Time-Out Control Command and Status Register (PRTOCCSR)

PRTOCCSR fields are described in Table 16-23.

Table 16-23. PRTOCCSR Field Descriptions

Bitws	Name	Description
0–23	TV	Time-out value
24–31	—	Reserved

16.3.2.19 Port General Control Command and Status Register (PGCCSR)

The PGCCSR is shown in Figure 16-21.

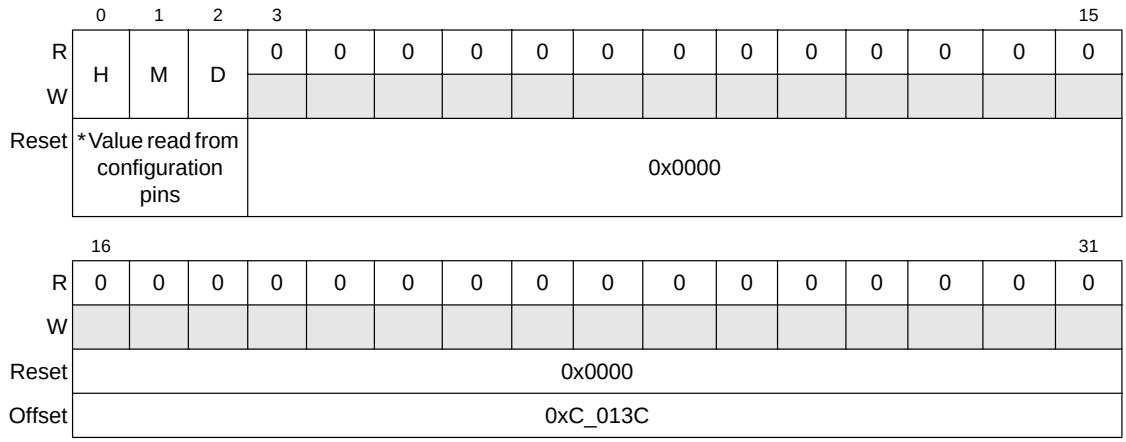


Figure 16-21. Port General Control Command and Status Register (PGCCSR)

PGCCSR fields are described in Table 16-24.

Table 16-24. 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

16.3.2.20 Port Link Maintenance Request Command and Status Register (PLMREQCSR)

The PLMREQCSR is shown in Figure 16-22.

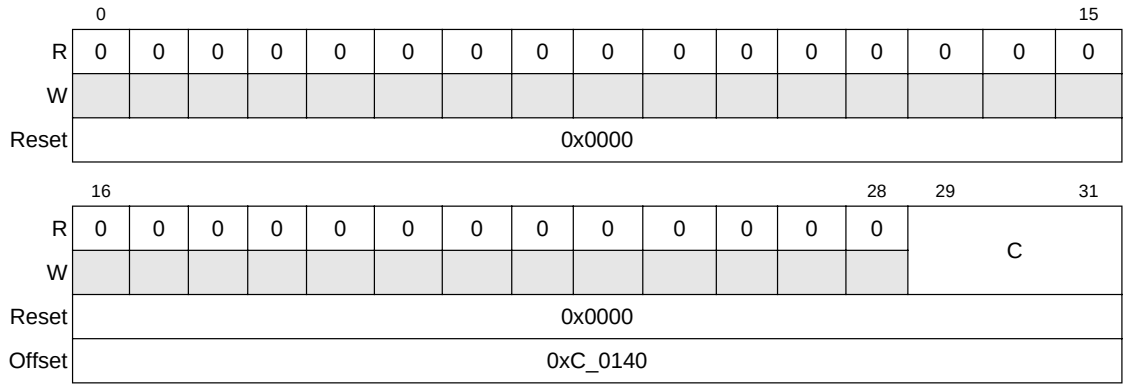


Figure 16-22. Port Link Maintenance Request Command and Status Register (PLMREQCSR)

PLMREQCSR fields are described in Table 16-25.

Table 16-25. PLMREQCSR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29–31	C	LINK_REQUEST command to send

16.3.2.21 Port Link Maintenance Response Command and Status Register (PLMRESPCSR)

The PLMRESPCSR, shown in Figure 16-23, is a read only register.

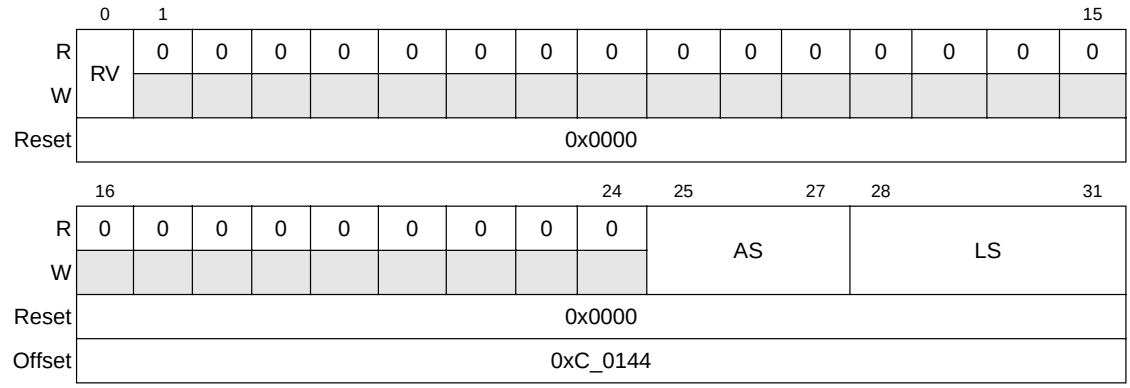


Figure 16-23. Port Link Maintenance Response Command and Status Register (PLMRESPCSR)

PLMRESPCSR fields are described in Table 16-26.

Table 16-26. 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

16.3.2.22 Port Local AckID Status Command and Status Register (PLASCSR)

The PLASCSR is shown in Figure 16-24.

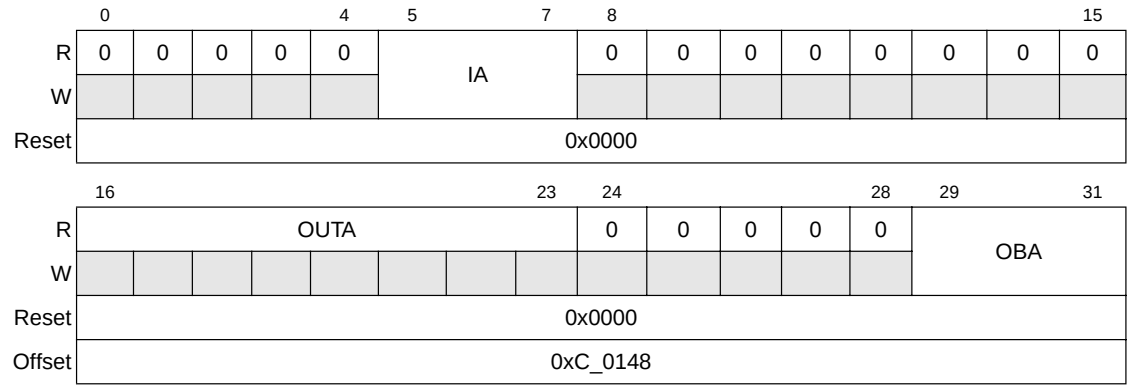


Figure 16-24. Port Local AckID Status Command and Status Register (PLASCSR)

PLASCSR fields are described in Table 16-27.

Table 16-27. 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.

16.3.2.23 Port Error and Status Command and Status Register (PESCSR)

The PESCSR is shown in Figure 16-25.

	0										10	11	12	13	14	15
R	0	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 16-25. Port Error and Status Command and Status Register (PESCSR)

PESCSR fields are described in Table 16-28.

Table 16-28. 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 16-28. 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.

16.3.2.24 Port Control Command and Status Register (PCCSR)

The PCCSR is shown in Figure 16-26.

	0	1	2	3	4	5	6	7	8	9	10					15
R	OPW	OPE	OPD	0	IPW	IPE	IPD	0	ECD	TOD	0	0	0	0	0	0
W																
Reset	0x4400															

	16															31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0x0000															
Offset	0xC_015C															

Figure 16-26. Port Control Command and Status Register (PCCSR)

PCCSR fields are described in Table 16-29.

Table 16-29. 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 with 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

16.3.3 Implementation Registers

Following are the RapidIO implementation registers.

16.3.3.1 General Registers

16.3.3.1.1 Configuration Register (CR)

The CR is shown in Figure 16-27.

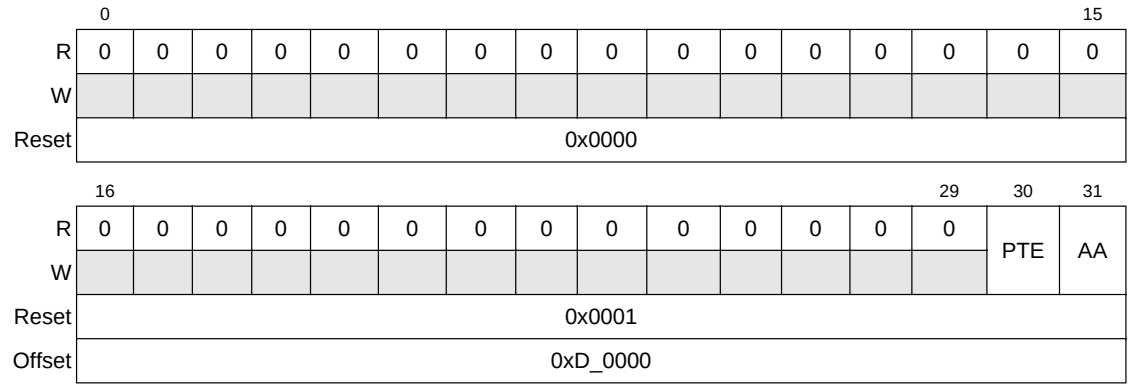


Figure 16-27. Configuration Register (CR)

CR fields are described in Table 16-30.

Table 16-30. 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 recieving 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.

16.3.3.1.2 Port Configuration Register (PCR)

The PCR is shown in Figure 16-28.

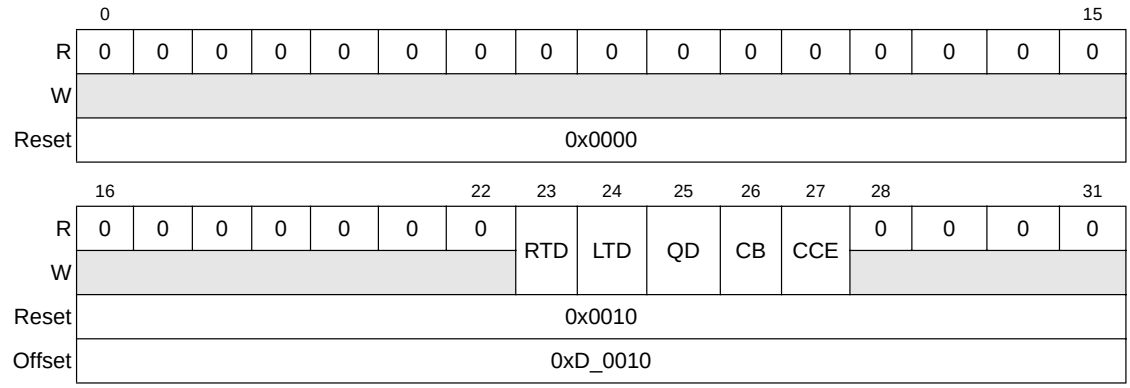


Figure 16-28. Port Configuration Register (PCR)

PCR fields are described in Table 16-31.

Table 16-31. 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

16.3.3.1.3 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 16-29.

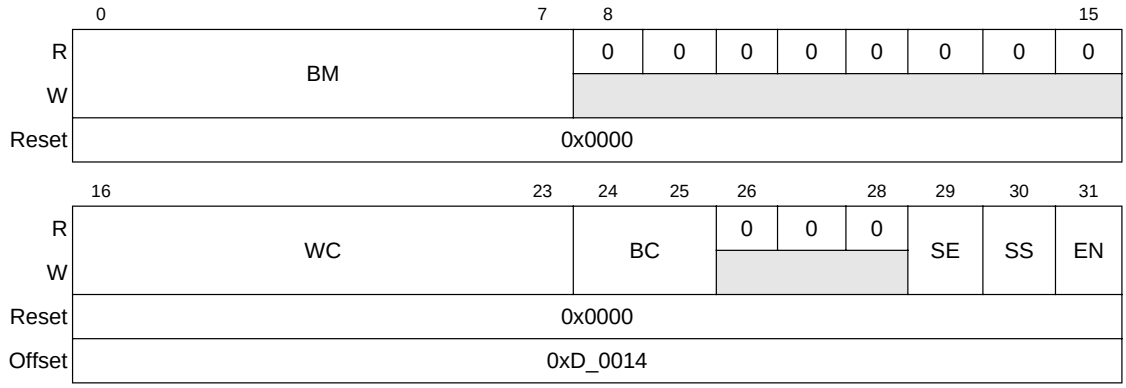


Figure 16-29. Port Error Injection Register (PEIR)

PEIR fields are described in Table 16-32.

Table 16-32. 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.

16.3.3.2 ATMU Registers

Although, the RapidIO Specification allows for 48 and 64-bit addresses, the MPC8560 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.

16.3.3.2.1 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 16-30.

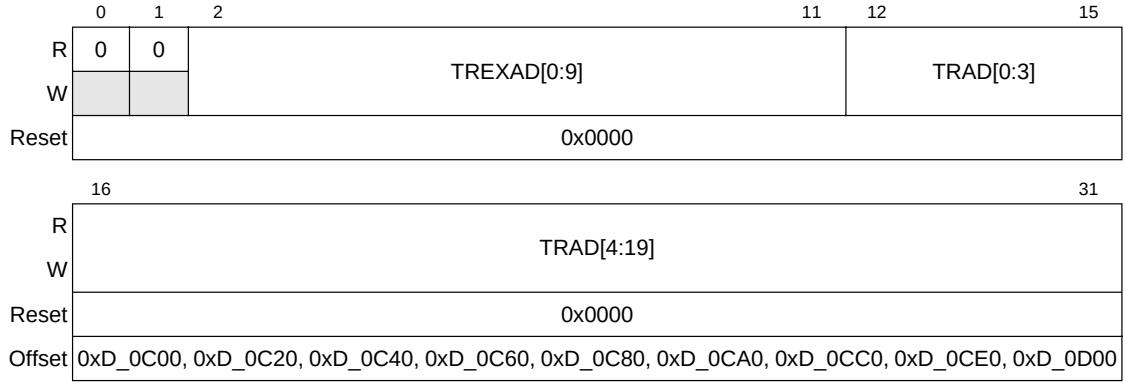


Figure 16-30. RapidIO Outbound Window Translation Address Registers 0–8 (ROWTAR*n*)

ROWTARs fields are described in Table 16-33.

Table 16-33. 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 0–7of 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 16-31 below.

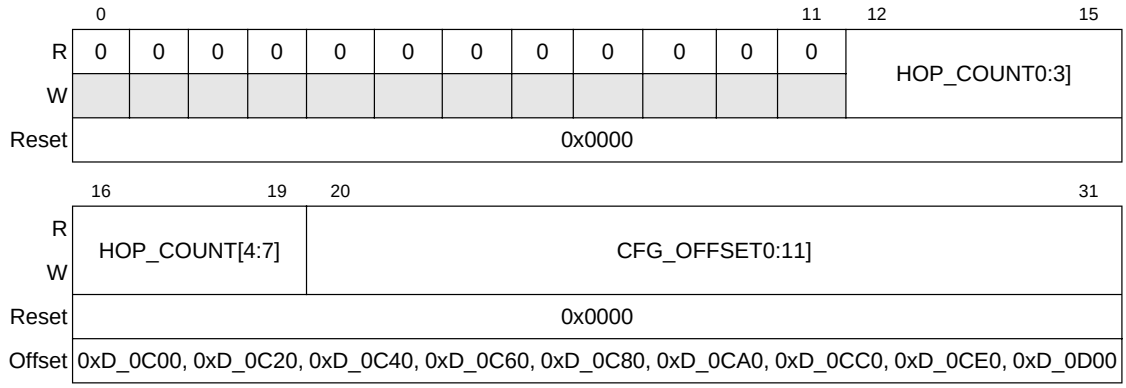


Figure 16-31. RapidIO Outbound Window Translation Address Registers 0–8 for Maintenance Transactions (ROWTAR*n*)

Table 16-34 shows the field descrptions 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 16-34. 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.

16.3.3.2.2 RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR*n*)

The RapidIO outbound window translation address registers (ROWBAR*n*s) 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 16-32.

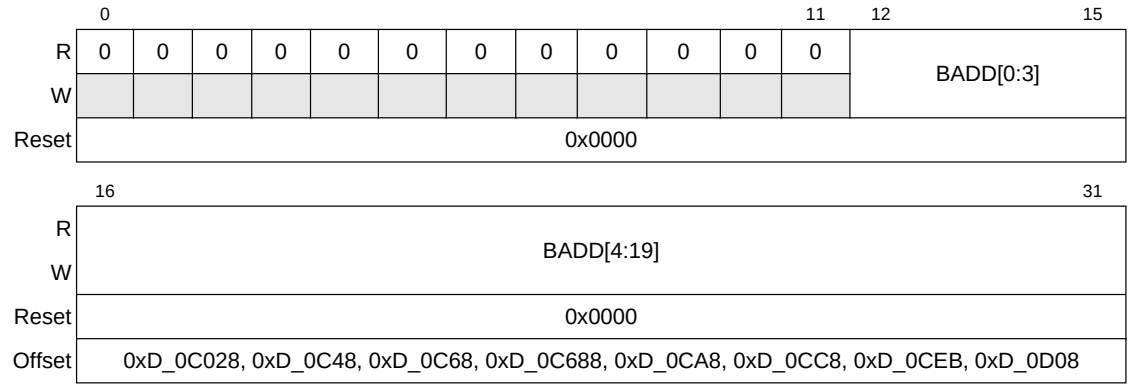


Figure 16-32. RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR*n*)

ROWBAR*n*s fields are described in Table 16-35.

Table 16-35. 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.

16.3.3.2.3 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 16-33.

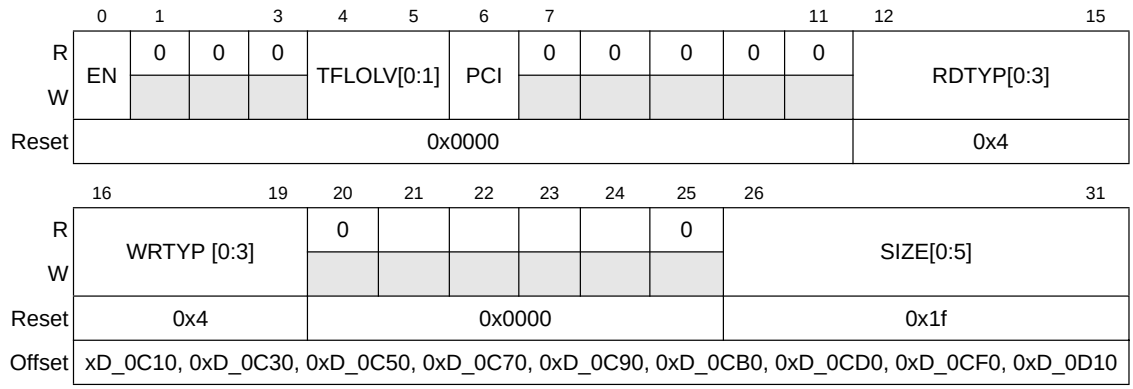


Figure 16-33. RapidIO Outbound Window Attributes Registers 0–8 (ROWAR*n*)

ROWARs fields are described in Table 16-36.

Table 16-36. 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 16.7.1.2, “Outbound Special Transactions and Requirements,” on page 16-113 for more information on MAINTENANCE transactions.
16–19	WRTYP	Write transaction typ.e. 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 16.7.1.2, “Outbound Special Transactions and Requirements,” on page 16-113 for more information on MAINTENANCE and DOORBELL transactions.

Table 16-36. 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

16.3.3.2.4 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 16-34.

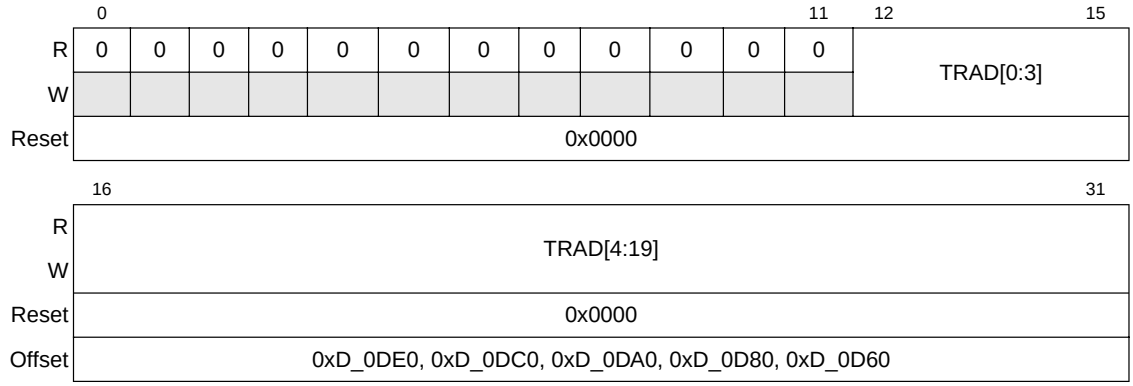


Figure 16-34. RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR*n*)

RIWTARs fields are described in Table 16-37.

Table 16-37. 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.

16.3.3.2.5 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 16-35.

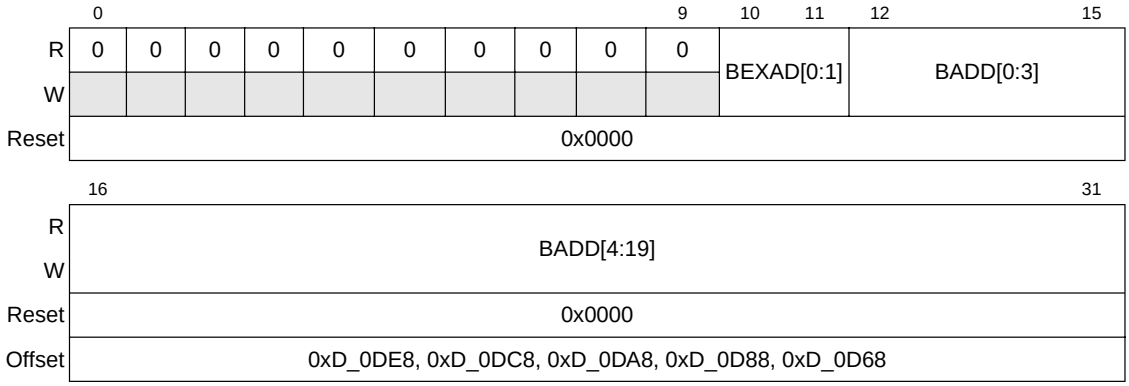


Figure 16-35. RapidIO Inbound Window Base Address Registers 1–4 (RIWBAR*n*)

RIWBARs fields are described in Table 16-38.

Table 16-38. 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.

16.3.3.2.6 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 16-36.

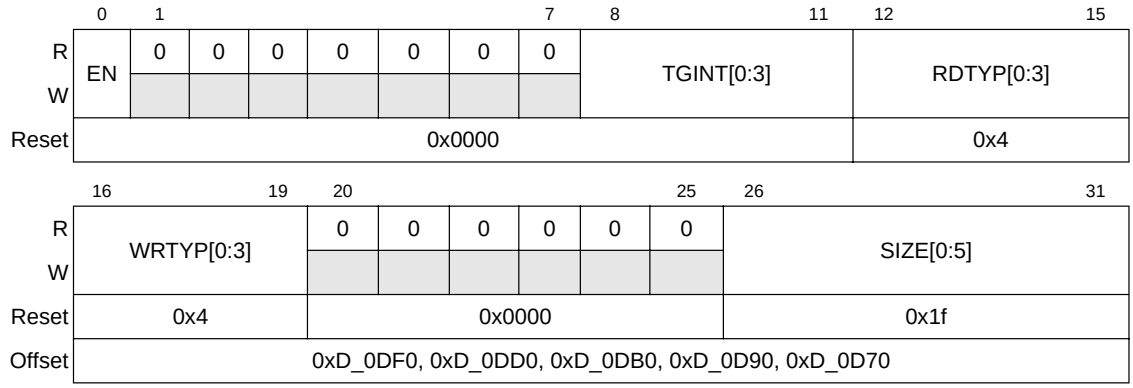


Figure 16-36. RapidIO Inbound Window Attributes Registers 0–4 (RIWAR*n*)

RIWARs fields are described in Table 16-39.

Table 16-39. 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 16-39. 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 processo 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 *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

16.3.3.3 Error Management Registers

16.3.3.3.1 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 16-37.

	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	URSSE	URESE	URSDE	MERE	MDFE
W																
Reset	0x0000															
Offset	0xD_0E00															

Figure 16-37. Port Notification/Fatal Error Detect Register (PNFEDR)

PNFEDR fields are described in Table 16-40.

Table 16-40. 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 segement 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 detedted 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 16-40. 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 unsolicted 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

16.3.3.3.2 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 16-38.

	0	1	2			5	6	7	8	9	10	11	12	13	14	15
R	0	AXD	0	0	0	0	ITD	MSD	NTID	SD	DSD	BMD	ETD	RTD	RERD	LTD
W																
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	IRSFD	IRTD	BRRD	BW RD	URS SD	URE SD	URSDD	MERD	MD FD
W																
Reset	0x0000															
Offset	0xD_0E04															

Figure 16-38. Port Notification/Fatal Error Disable Register (PNFEDiR)

PNFEDiR fields are described in Table 16-41.

Table 16-41. 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 16-41. 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 16-41. 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	URESD	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

16.3.3.3.3 Port Notification/Fatal Error Interrupt Enable Register (PNFEIER)

The PNFEIER, shown in Figure 16-39, 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	AXIE	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	NAIE	IRFIE	URDIE	URIE	URSIE	IWRIE	IRSFIE	IRTIE	BRRIE	BWRIE	URSSIE	URSIE	URSDIE	MERIE	MDFIE
W																
Reset	0x0000															
Offset	0xD_0E08															

Figure 16-39. Port Notification/Fatal Error Interrupt Enable Register (PNFEIER)

PNFEIER fields are described in Table 16-42.

Table 16-42. PNFEIER Field Descriptions

Bits	Name	Description
0	—	Reserved
1	AXIE	ATMU window crossing error interrupt enable
2–5	—	Reserved

Table 16-42. PNFEIER Field Descriptions (continued)

Bits	Name	Description
6	ITIE	Illegal Transaction Target Error interrupt enable
7	MSIE	Message size error interrupt enable
8	NTIIE	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

16.3.3.3.4 Port Error Control Register (PECR)

The PECR, shown in Figure 16-40, indicates that a valid packet has been captured when a notification or fatal error has occurred.

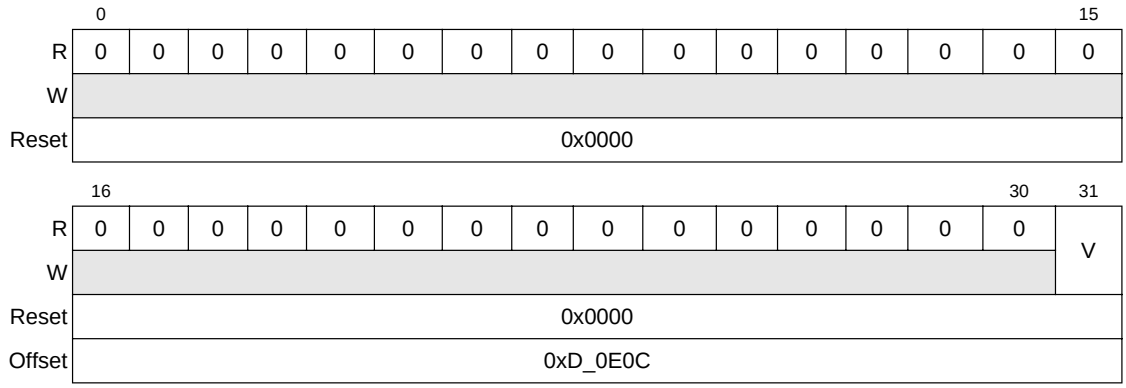


Figure 16-40. Port Error Control Register (PECR)

PECR fields are described in Table 16-43.

Table 16-43. PECR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	V	Packet registers contain valid information.

16.3.3.3.5 Port Error Packet/Control Symbol Register 0 (PEPCSR0)

The port error packet/control symbol register 0 (PEPCSR0), shown in Figure 16-41, 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.

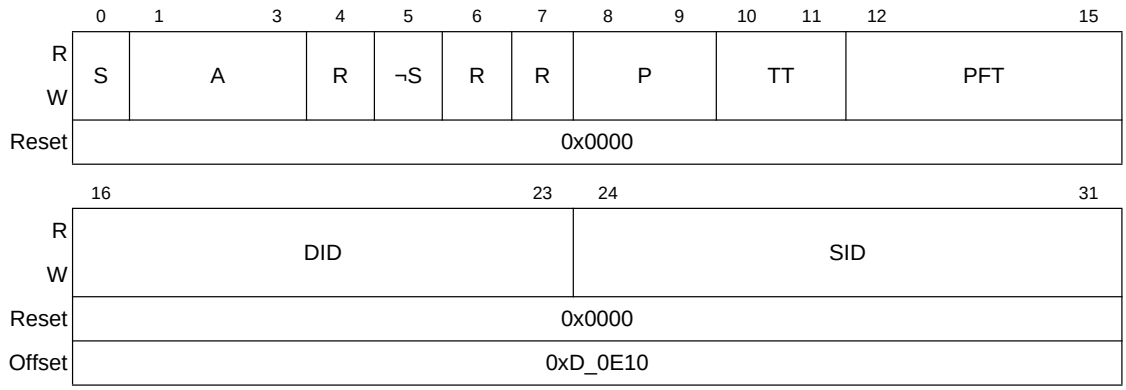


Figure 16-41. Port Error Packet/Control Symbol Register 0 (PEPCSR0)

PEPCSR0 fields are described in Table 16-44.

Table 16-44. 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

16.3.3.3.6 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 16.3.3.3.7, “RapidIO Packet Error Capture Register 1, Type 1 (PECR1T1),” through Section 16.3.3.3.15, “RapidIO Packet Error Capture Register 1, Type 13 (PECR1T13).”

16.3.3.3.7 RapidIO Packet Error Capture Register 1, Type 1 (PECR1T1)

Figure 16-42 displays the RapidIO packet error capture register 1 for a type 1 packet format (PECR1T1).

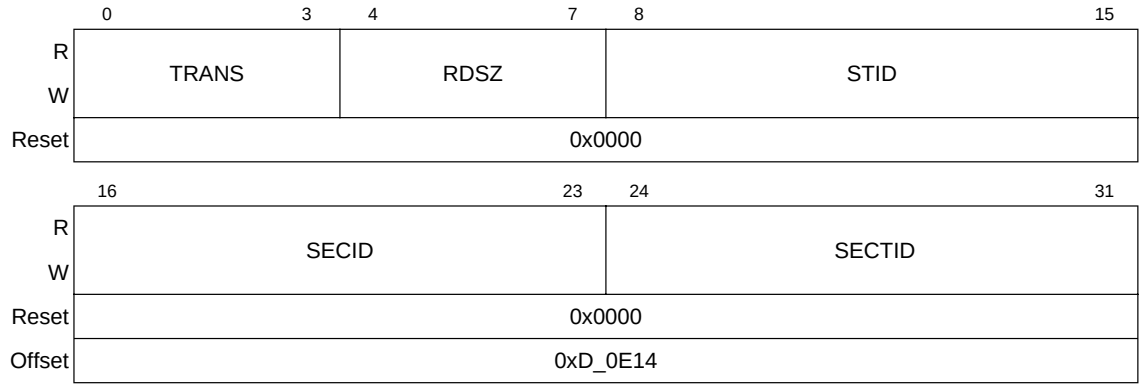


Figure 16-42. RapidIO Packet Error Capture Register 1, Type 1 (PECR1T1)

PECR1T1 fields are described in Table 16-45.

Table 16-45. 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

16.3.3.3.8 RapidIO Packet Error Capture Register 1, Type 2 (PECR1T2)

Figure 16-43 displays the RapidIO packet error capture register 1 for a type 2 packet format (PECR1T2).

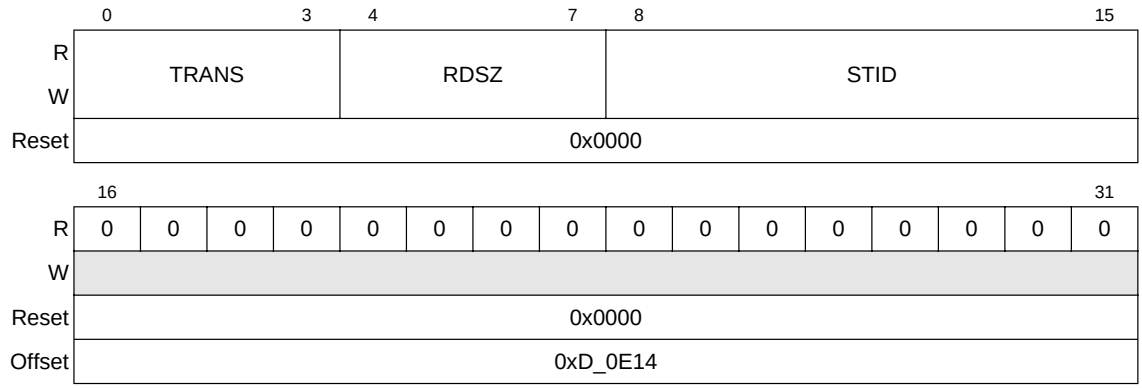


Figure 16-43. RapidIO Packet Error Capture Register 1, Type 2 (PECR1T2)

PECR1T2 fields are described in Table 16-46.

Table 16-46. 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

16.3.3.3.9 RapidIO Packet Error Capture Register 1, Type 5 (PECR1T5)

Figure 16-44 displays the RapidIO packet error capture register 1 for a type 5 packet format (PECR1T5).

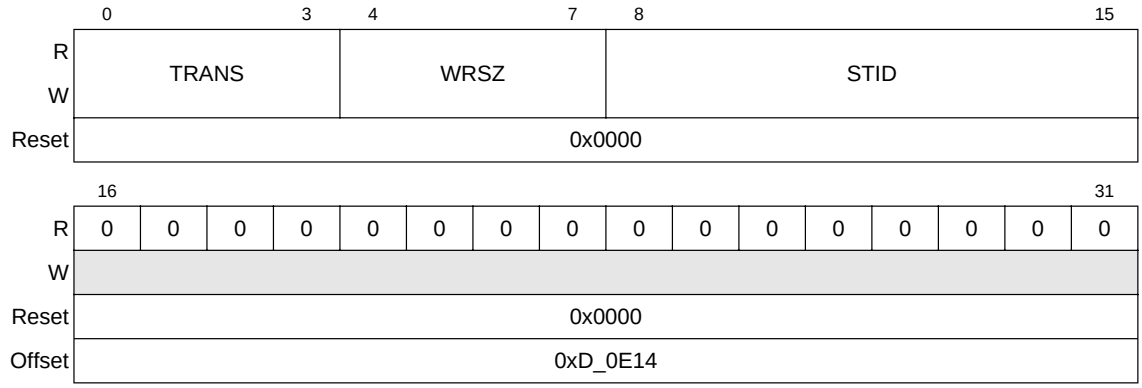


Figure 16-44. RapidIO Packet Error Capture Register 1, Type 5 (PECR1T5)

PECR1T5 fields are described in Table 16-47.

Table 16-47. 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

16.3.3.3.10 RapidIO Packet Error Capture Register 1, Type 6 (PECR1T6)

Figure 16-45 displays the RapidIO packet error capture register 1 for a type 6 packet format (PECR1T6).

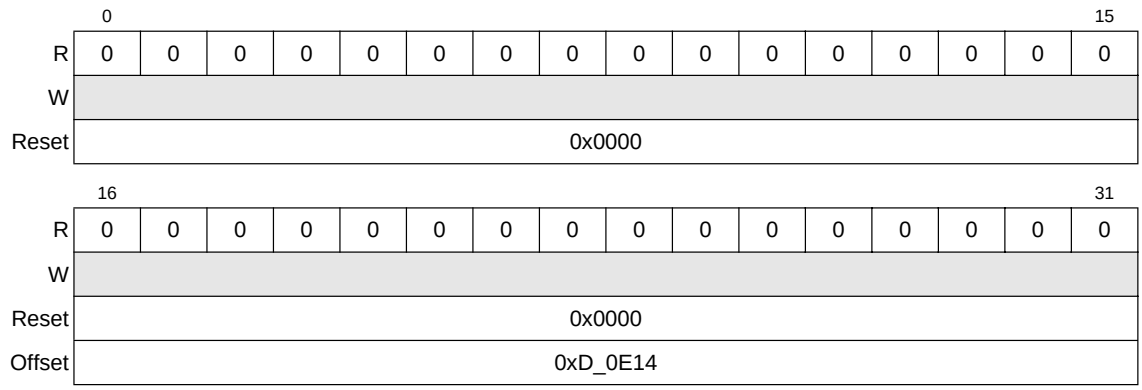


Figure 16-45. RapidIO Packet Error Capture Register 1, Type 6 (PECR1T6)

PECR1T6 fields are described in Table 16-48.

Table 16-48. PECR1T6 Field Descriptions

Bits	Name	Description
0–31	—	Reserved

16.3.3.3.11 RapidIO Packet Error Capture Register 1, Type 8 Request (PECR1T8RQ)

Figure 16-46 displays the RapidIO packet error capture register 1 for a type 8 request packet format (PECR1T8RQ).

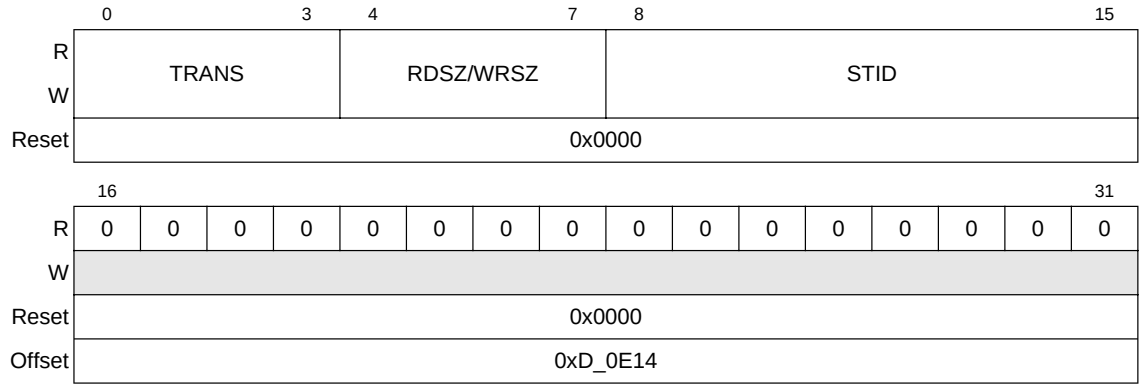


Figure 16-46. RapidIO Packet Error Capture Register 1, Type 8 Request (PECR1T8RQ)

PECR1T8RQ fields are described in Table 16-49.

Table 16-49. 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

16.3.3.3.12 RapidIO Packet Error Capture Register 1, Type 8 Response (PECR1T8R)

Figure 16-47 displays the RapidIO packet error capture register 1 for a type 8 response packet format (PECR1T8R).



Figure 16-47. RapidIO Packet Error Capture Register 1, Type 8 Response (PECR1T8R)

PECR1T8R fields are described in Table 16-50.

Table 16-50. 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

16.3.3.3.13 RapidIO Packet Error Capture Register 1, Type 10 (PECR1T10)

Figure 16-48 displays the RapidIO packet error capture register 1 for a type 10 packet format (PECR1T10).



Figure 16-48. RapidIO Packet Error Capture Register 1, Type 10 (PECR1T10)

PECR1T10 fields are described in Table 16-51.

Table 16-51. PECR1T10 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	STID	Source transaction ID
16–31	—	Reserved

16.3.3.3.14 RapidIO Packet Error Capture Register 1, Type 11 (PECR1T11)

Figure 16-49 displays the RapidIO packet error capture register 1 for a type 11 packet format (PECR1T11).

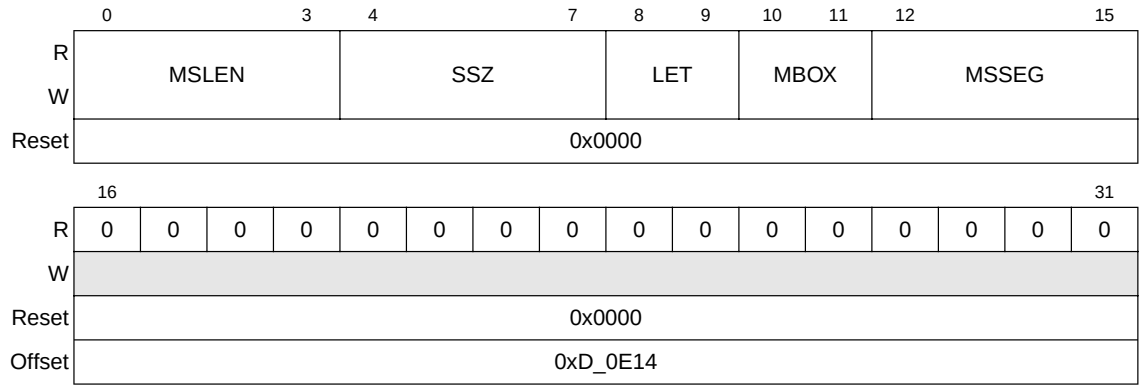


Figure 16-49. RapidIO Packet Error Capture Register 1, Type 11 (PECR1T11)

PECR1T11 fields are described in Table 16-52.

Table 16-52. 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

16.3.3.3.15 RapidIO Packet Error Capture Register 1, Type 13 (PECR1T13)

Figure 16-50 displays the RapidIO packet error capture register 1 for a type 13 packet format (PECR1T13).



Figure 16-50. RapidIO Packet Error Capture Register 1, Type 13 (PECR1T13)

PECR1T13 fields are described in Table 16-53.

Table 16-53. 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

16.3.3.3.16 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 16.3.3.3.17 through 16.3.3.3.22.

16.3.3.3.17 RapidIO Packet Error Capture Register 2, Types 1, 2, and 5 (PECR2T125)

Figure 16-51 displays the RapidIO packet error capture register 2 for a type 1, 2, and 5 packet format (PECR2T125).

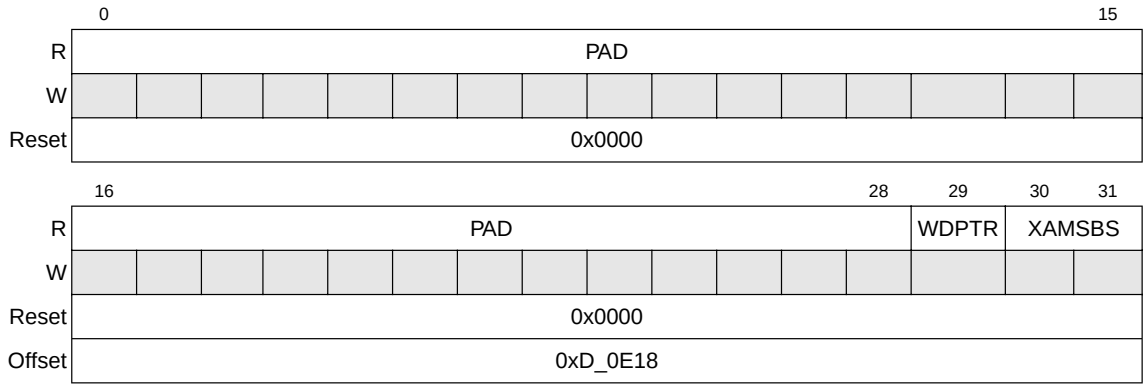


Figure 16-51. RapidIO Packet Error Capture Register 2, Type 1 (PECR2T125)

PECR2T125 fields are described in Table 16-54.

Table 16-54. 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

16.3.3.3.18 RapidIO Packet Error Capture Register 2, Type 6 (PECR2T6)

Figure 16-52 displays the RapidIO packet error capture register 2 for a type 6 packet format (PECR2T6).

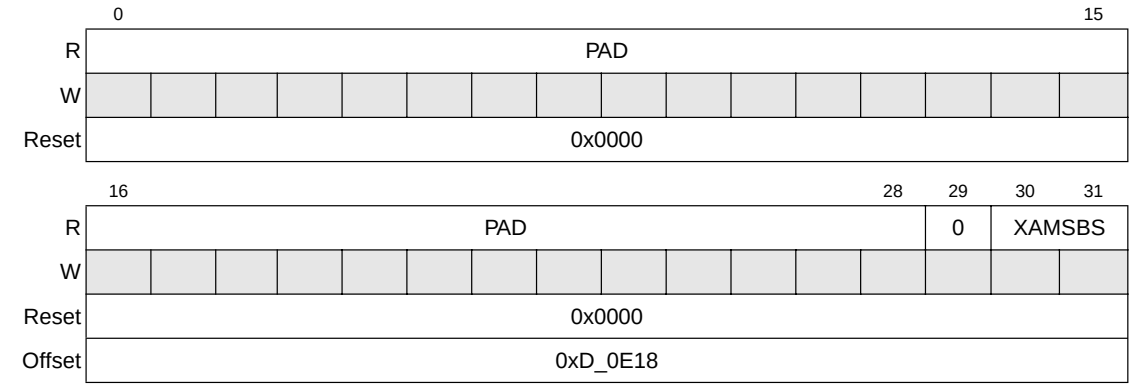


Figure 16-52. RapidIO Packet Error Capture Register 2, Type 6 (PECR2T6)

PECR2T6 fields are described in Table 16-55.

Table 16-55. 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

16.3.3.3.19 RapidIO Packet Error Capture Register 2, Type 8 (PECR2T8)

Figure 16-53 displays the RapidIO packet error capture register 2 for a type 8 packet format (PECR2T8).

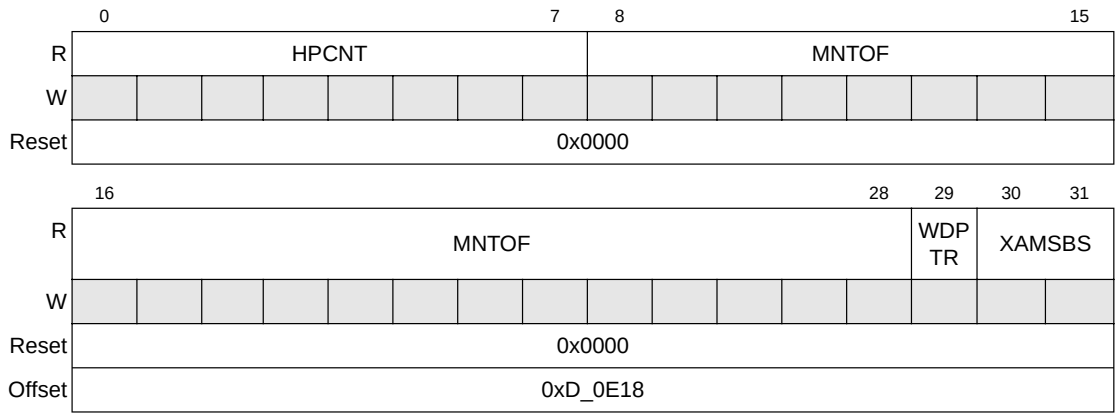


Figure 16-53. RapidIO Packet Error Capture Register 2, Type 8 (PECR2T8)

PECR2T8 fields are described in Table 16-56.

Table 16-56. 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

16.3.3.3.20 RapidIO Packet Error Capture Register 2, Type 8 Response (PECR2T8R)

Figure 16-54 displays the RapidIO packet error capture register 2 for a type 8 response packet format (PECR2T8R).

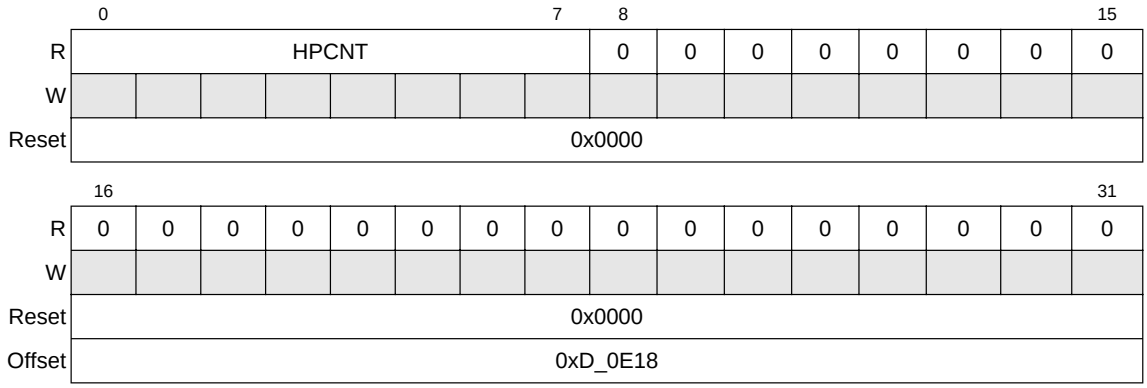


Figure 16-54. RapidIO Packet Error Capture Register 2, Type 8 Response (PECR2T8R)

PECR2T8R fields are described in Table 16-57.

Table 16-57. PECR2T8R Field Descriptions

Bits	Name	Description
0–7	HPCNT	Hop count field of a packet
8–31	—	Reserved

16.3.3.3.21 RapidIO Packet Error Capture Register 2, Type 10 (PECR2T10)

Figure 16-55 displays the RapidIO packet error capture register 2 for a type 10 packet format (PECR2T10).

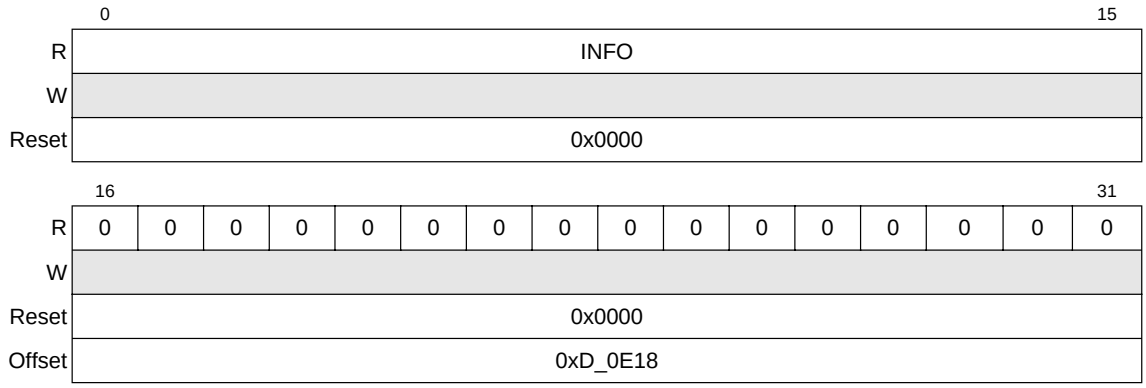


Figure 16-55. RapidIO Packet Error Capture Register 2, Type 10 (PECR2T10)

PECR2T10 fields are described in Table 16-58.

Table 16-58. PECR2T10 Field Descriptions

Bits	Name	Description
0–15	INFO	Information field of a packet
16–31	—	Reserved

16.3.3.3.22 RapidIO Packet Error Capture Register 2, Types 11 and 13 (PECR2T11)

Figure 16-56 displays the RapidIO packet error capture register 2 for types 11 and 13 packet formats (PECR2T11).



Figure 16-56. RapidIO Packet Error Capture Register 2, Type 11 (PECR2T11)

PECR2T11 fields are described in Table 16-59.

Table 16-59. PECR2T11 Field Description

Bits	Name	Description
0–31	—	Reserved

16.3.3.3.23 Port Recoverable Error Detect Register (PREDR)

The PREDR, shown in Figure 16-57, 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	INA PS	INA BC	INASE	INAGE	URE	UAE	ABRE	UTE	USE	ULE	UEE	LRE	FAE	FTEE	ATE
W																
Reset	0x0000															
Offset	0xD_0E20															

Figure 16-57. Port Recoverable Error Detect Register (PREDR)

PREDR fields are described in Table 16-60.

Table 16-60. 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 16-60. 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 16-60. 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)

16.3.3.3.24 Port Error Recovery Threshold Register (PERTR)

The PERTR is shown in Figure 16-58.

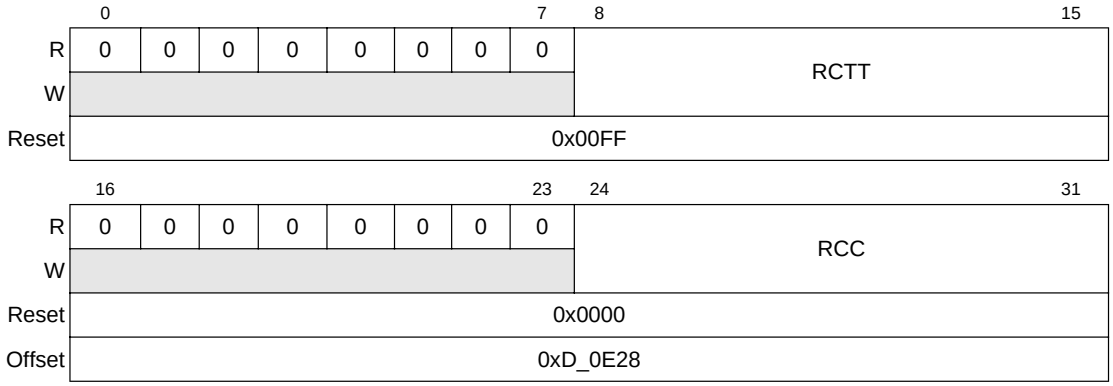


Figure 16-58. Port Error Recovery Threshold Register (PERTR)

PERTR fields are described in Table 16-61.

Table 16-61. 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.

16.3.3.3.25 Port Retry Threshold Register (PRTR)

The PRTR is shown in Figure 16-59.

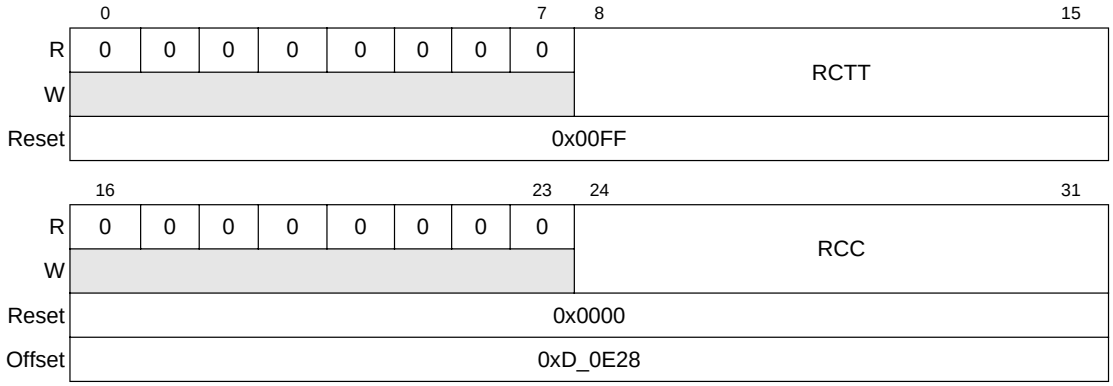


Figure 16-59. Port Retry Threshold Register (PRTR)

PRTR fields are described in Table 16-62.

Table 16-62. 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.

16.3.4 RapidIO Message Unit Registers

16.3.4.1 RapidIO Outbound Message Registers

The registers in this section control RapidIO outbound messages. The following provide partial descriptions of these registers.

16.3.4.1.1 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 16-60.

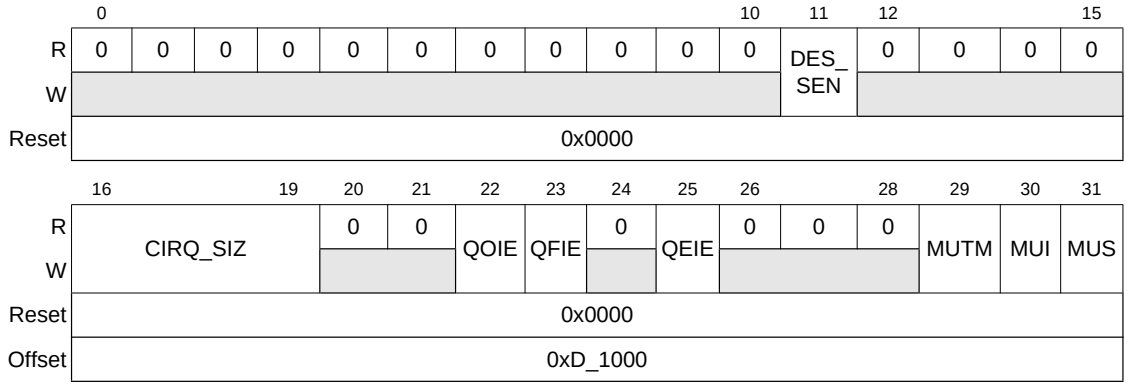


Figure 16-60. Outbound Mode Register (OMR)

OMR fields are described in Table 16-63.

Table 16-63. 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 16-63. 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.

16.3.4.1.2 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 16-61.

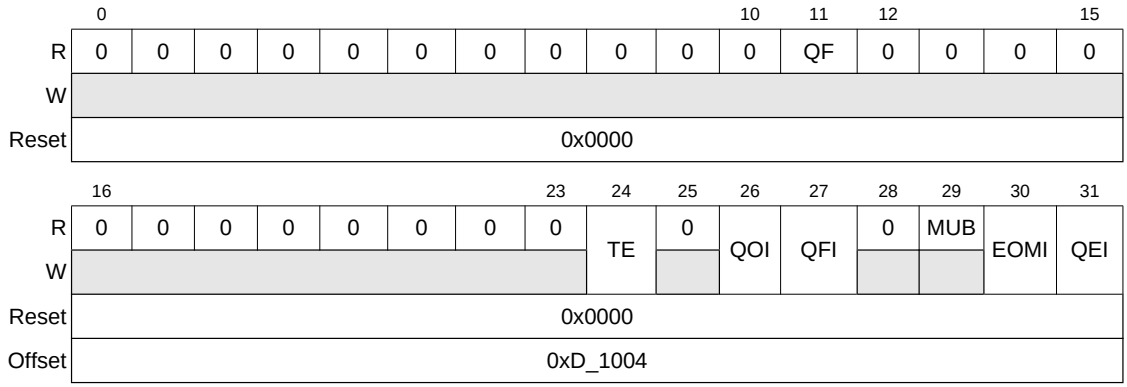


Figure 16-61. Outbound Status Register (OSR)

OSR fields are described in Table 16-64.

Table 16-64. 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 16-64. 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.

16.3.4.1.3 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 16-62.

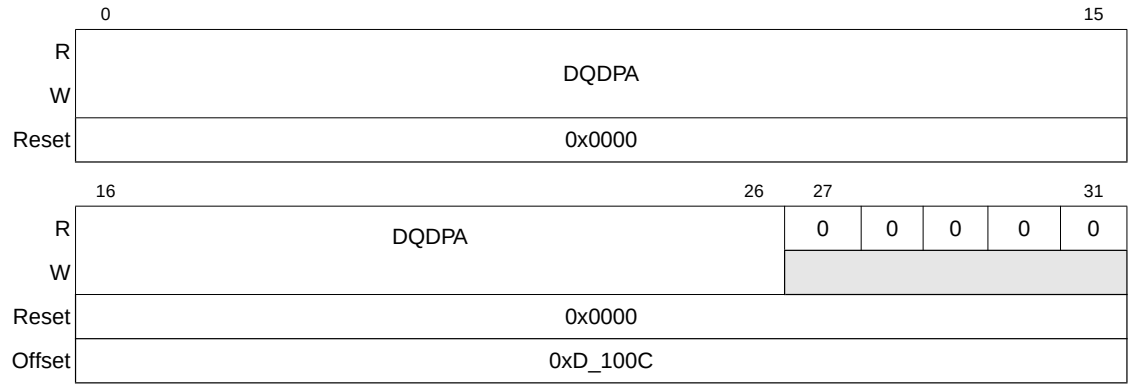


Figure 16-62. Outbound Descriptor Queue Dequeue Pointer Address Register (ODQDPAR)

ODQDPAR fields are described in Table 16-65.

Table 16-65. 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

16.3.4.1.4 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 16-63.

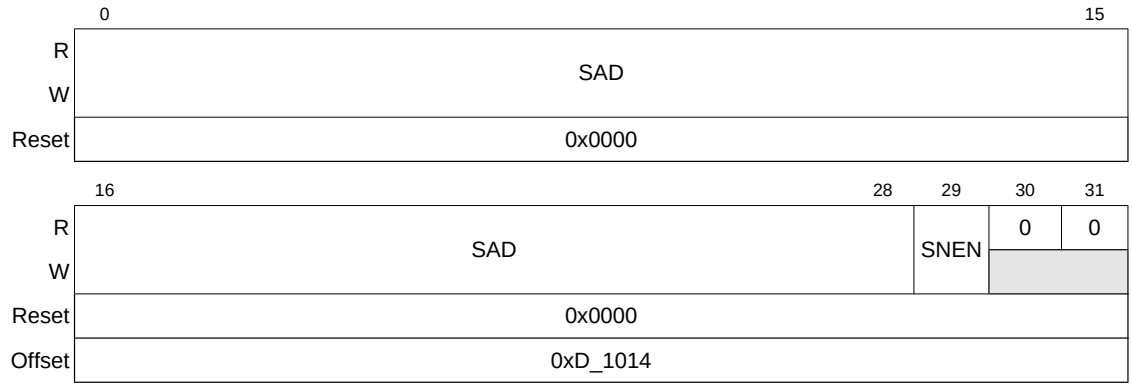


Figure 16-63. Outbound Unit Source Address Registers (OSAR)

OSAR fields are described in Table 16-66.

Table 16-66. 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

16.3.4.1.5 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 16-64.

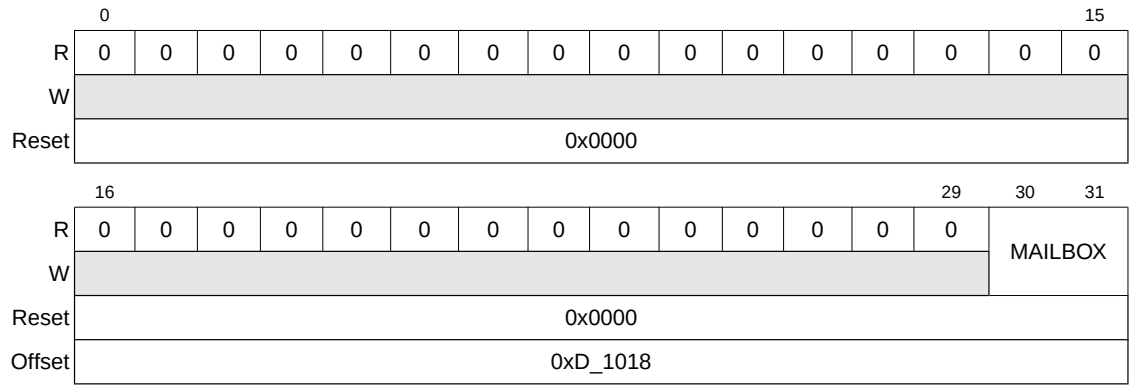


Figure 16-64. Outbound Destination Port Registers (ODPR)

ODPR fields are described in Table 16-67.

Table 16-67. ODPR Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30–31	MAILBOX	Value for mbox field in MESSAGE packet

16.3.4.1.6 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 16-65.

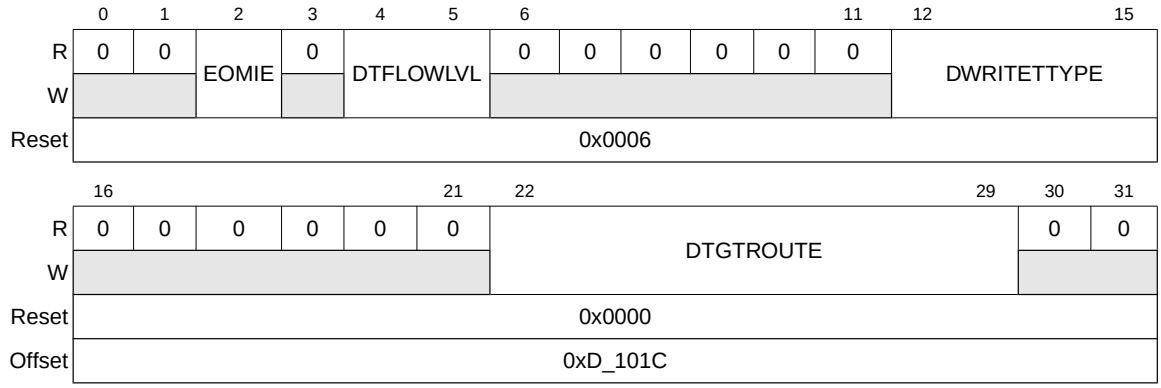


Figure 16-65. Outbound Destination Attributes Register (ODATR)

ODATR fields are described in Table 16-68.

Table 16-68. 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	DWRITETTYPE	Hardwired to 0110 (MESSAGE)
16–21	—	Reserved
22–29	DTGTROUTE	Destination target route. Contains the target route field of the transaction (Device ID of the target)
30–31	—	Reserved

16.3.4.1.7 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 16-66.

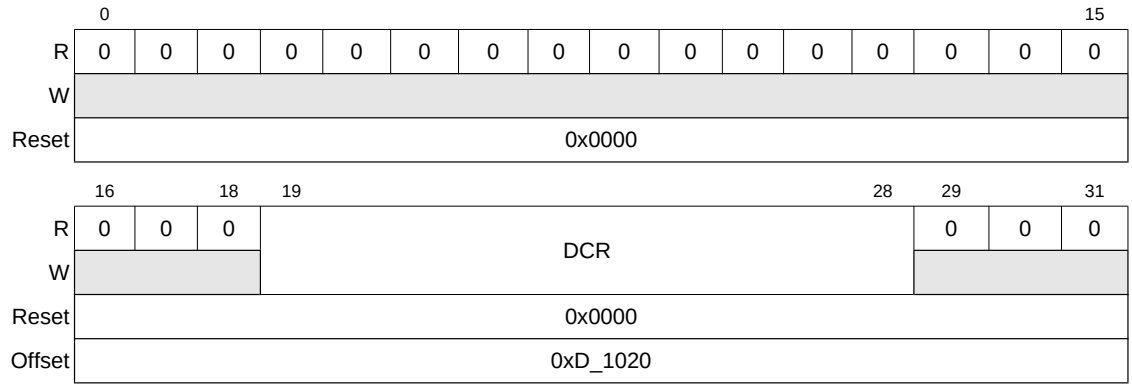


Figure 16-66. Outbound Double-word Count Register (ODCR)

ODCR fields are described in Table 16-69.

Table 16-69. 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

16.3.4.1.8 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 16-67.

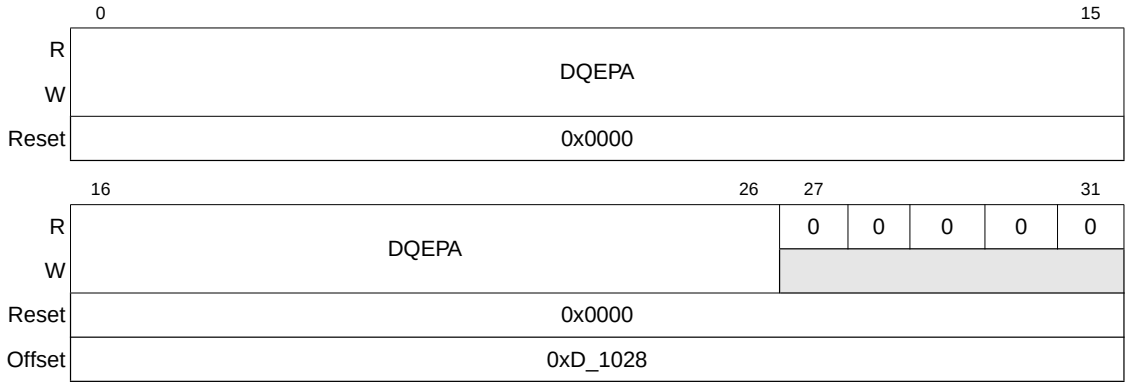


Figure 16-67. Outbound Descriptor Queue Enqueue Pointer Register (ODQEPAR)

ODQEPAR fields are described in Table 16-70.

Table 16-70. 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

16.3.4.2 RapidIO Inbound Message Registers

Registers in this section describe the RapidIO inbound message registers.

16.3.4.2.1 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 16-68.

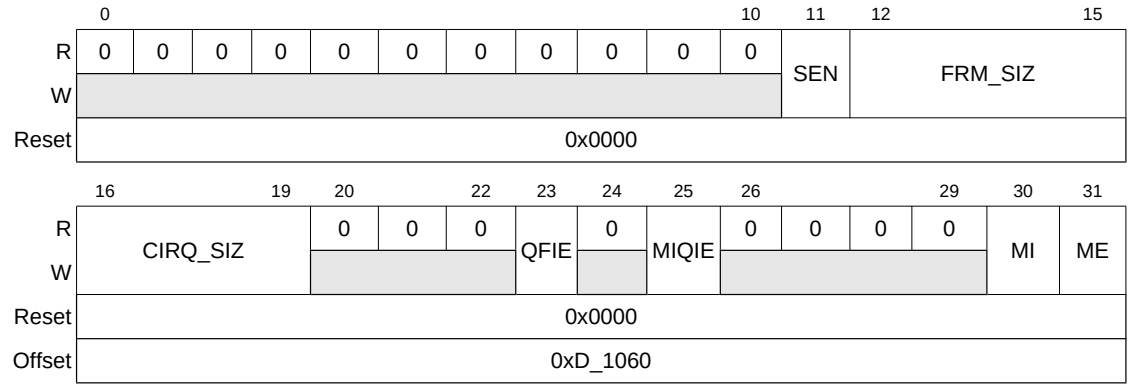


Figure 16-68. Inbound Mode Register (IMR)

IMR fields are described in Table 16-71.

Table 16-71. 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 16-71. 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.

16.3.4.2.2 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 16-69.

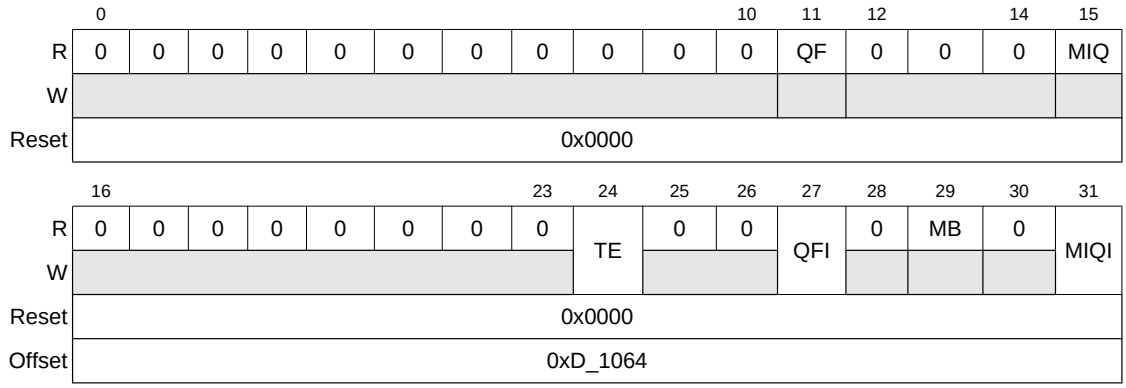


Figure 16-69. Inbound Status Register (ISR)

ISR fields are described in Table 16-72.

Table 16-72. 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)

16.3.4.2.3 Inbound Frame Queue Dequeue Pointer Address Register (IFQDPAR)

The inbound frame queue dequeue pointer address register (IFQDPAR) 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 IFQDPAR 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 IFQDPAR is shown in Figure 16-70.

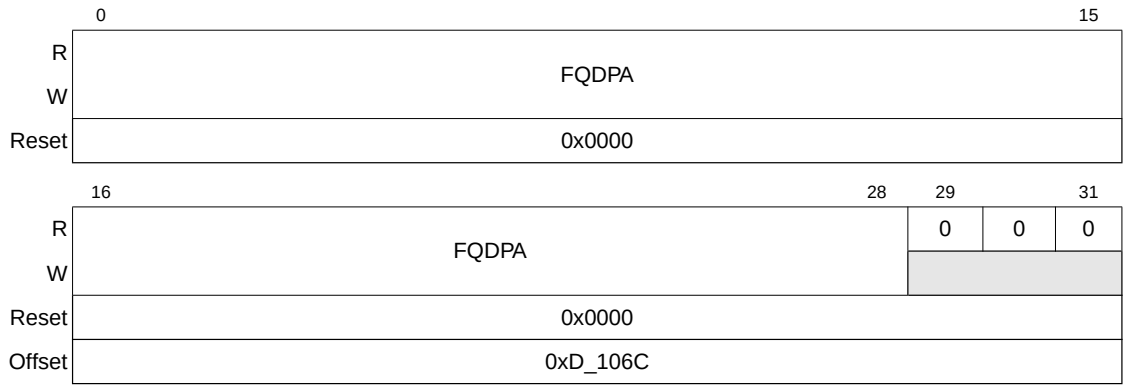


Figure 16-70. Inbound Frame Queue Dequeue Pointer Address Registers (IFQDPAR)

IFQDPAR fields are described in Table 16-73.

Table 16-73. IFQDPAR 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

16.3.4.2.4 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 16-71.

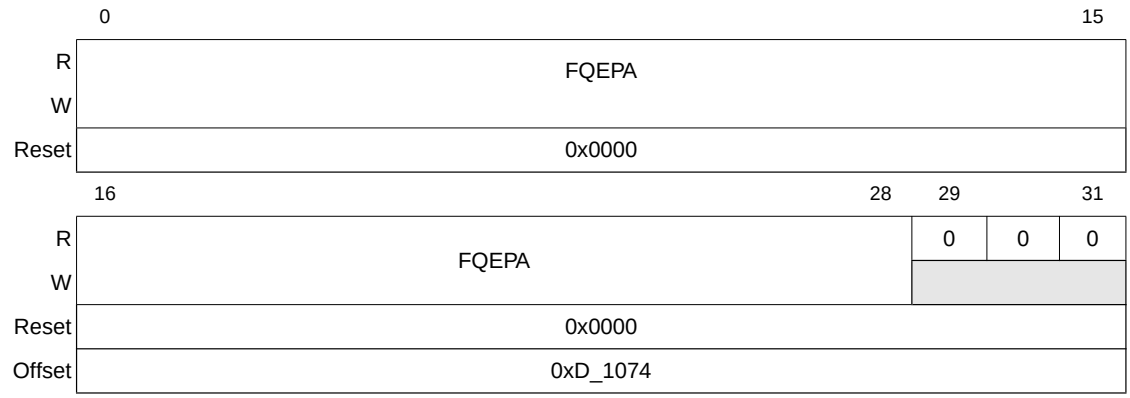


Figure 16-71. Inbound Frame Queue Enqueue Pointer Address Registers (IFQEPAR)

IFQEPAR fields are described in Table 16-74.

Table 16-74. 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

16.3.4.3 RapidIO Doorbell Registers

These registers control the RapidIO doorbell unit. The following sections provide descriptions of these registers.

16.3.4.3.1 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 16-72.

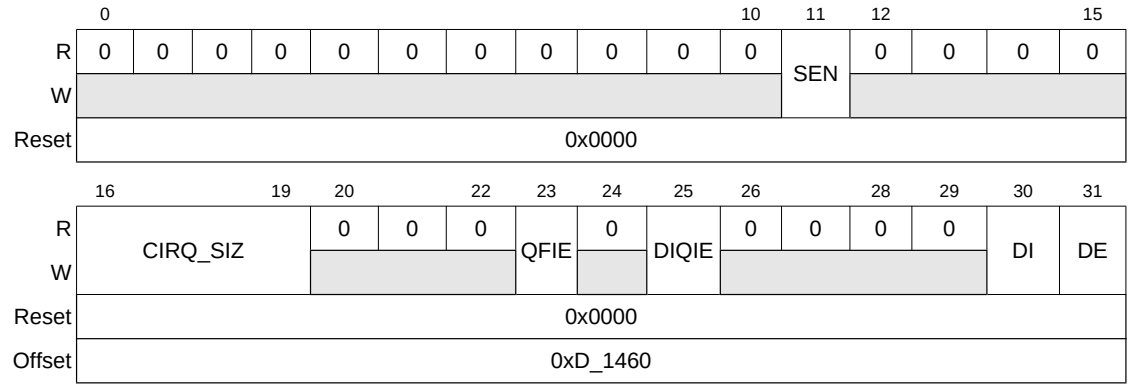


Figure 16-72. Doorbell Mode Register (DMR)

DMR fields are described in Table 16-75.

Table 16-75. 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 16-75. 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.

16.3.4.3.2 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 16-73.

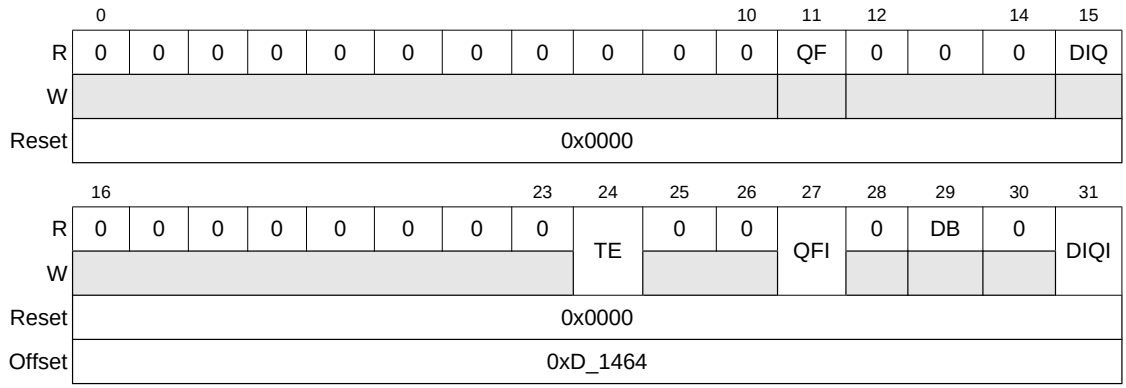


Figure 16-73. Doorbell Status Register (DSR)

DSR fields are described in Table 16-76.

Table 16-76. 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 16-76. 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)

16.3.4.3.3 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 16-74.

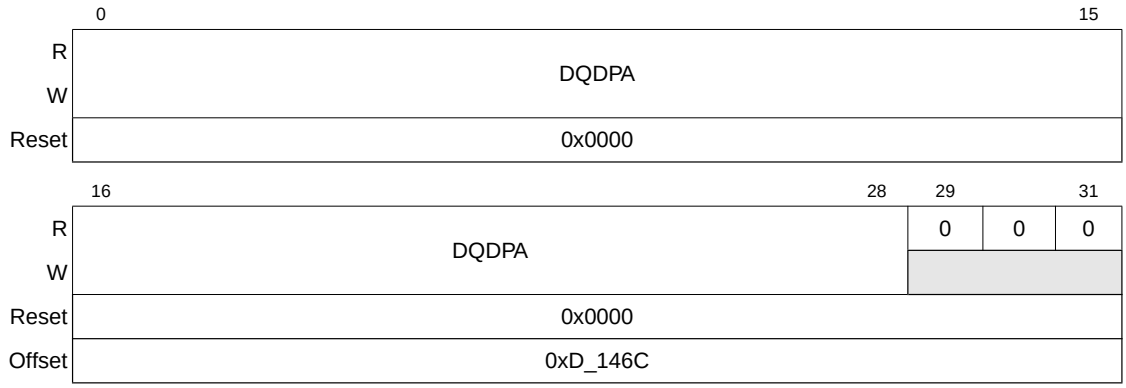


Figure 16-74. Doorbell Queue Dequeue Pointer Address Registers (DQDPAR)

DQDPAR fields are described in Table 16-77.

Table 16-77. DQDPAR 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

16.3.4.3.4 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 16-75.

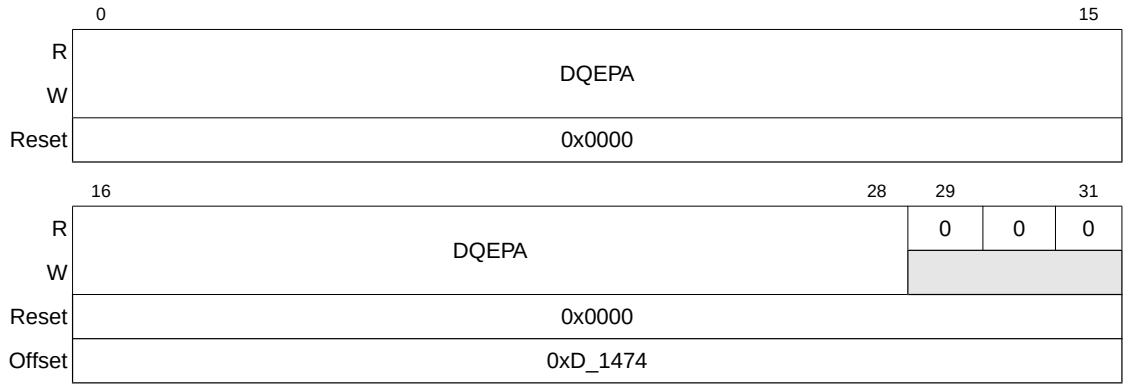


Figure 16-75. Doorbell Queue Enqueue Pointer Address Register (DQEPAR)

DQEPAR fields are described in Table 16-78.

Table 16-78. 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

16.3.4.4 RapidIO Port-Write Registers

These registers control the RapidIO port-write unit. The following sections provide partial descriptions of these registers.

16.3.4.4.1 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 16-76.

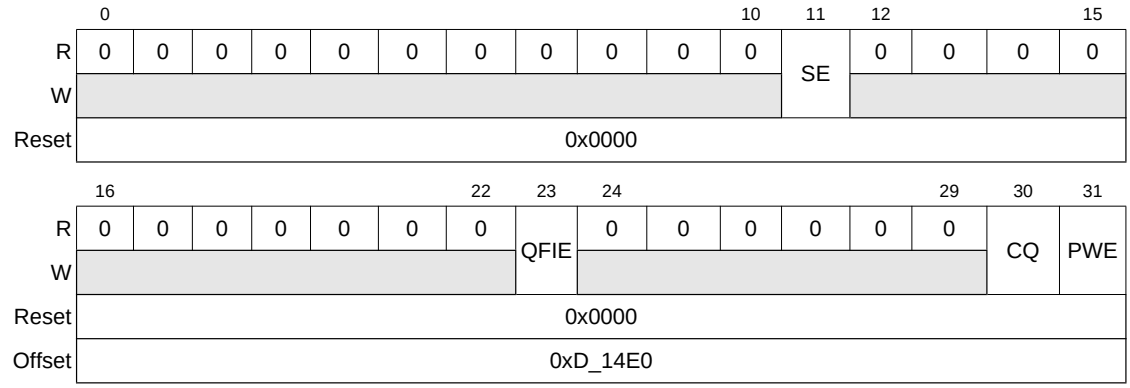


Figure 16-76. Port-Write Mode Register (PWMR)

PWMR fields are described in Table 16-79.

Table 16-79. 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.

16.3.4.4.2 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 16-77.

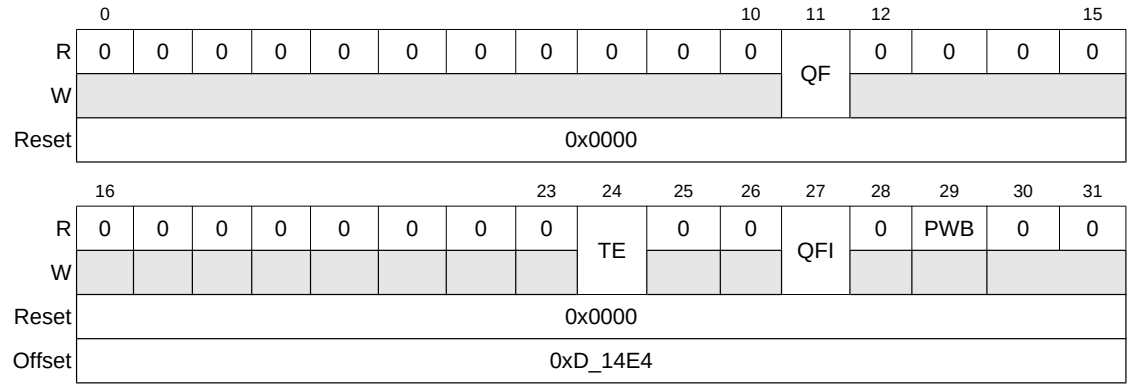


Figure 16-77. Port-Write Status Register (PWSR)

PWSR fields are described in Table 16-80.

Table 16-80. 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

16.3.4.4.3 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 16-78.

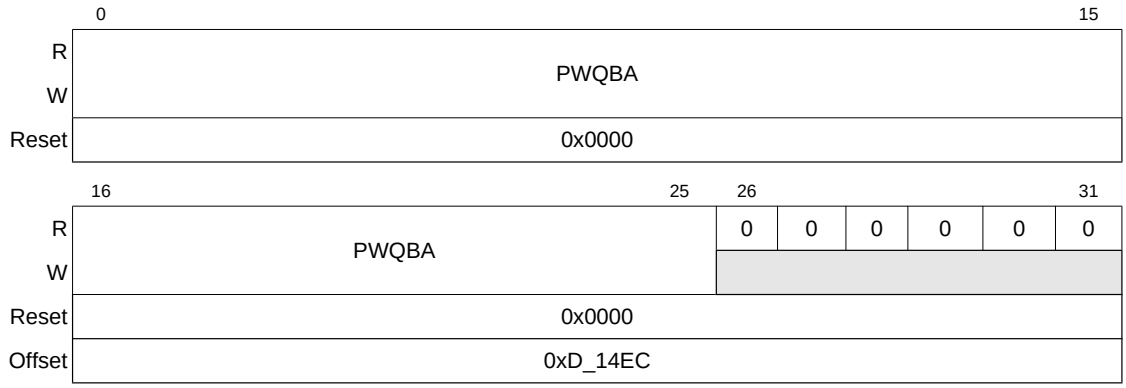


Figure 16-78. Port-Write Queue Base Address Register (PWQBAR)

PWQBAR fields are described in Table 16-81.

Table 16-81. 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

16.4 Functional Description

16.4.1 RapidIO Transaction, Packet, and Control Symbol Summary

Table 16-82 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 16-82. 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 16-82. 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 16-83 summarizes the packet formats for all the defined packet formats. The following definitions and Figure 16-79 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.

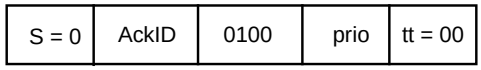


Figure 16-79. Physical Maintenance Field (PMF) Definition

Table 16-83. RapidIO Packet Format Summary (tt = 0b00)

Type	Packet Format
2 (0010)	Request To Home: PMF (12) | TYPE (4) | TARID (8) | SRCID (8) | TTYPE (4) | RDSIZE (4) | SRCTID (8) ADDRESS (29) | W (1) | ADDRESS (2) TTYPE:0010 (IO_READ_HOME); 0100 (NREAD); 1100 (ATOMIC INC); 1101 (ATOMIC DEC); 1110 (ATOMIC SET); 1111 (ATOMIC CLR
5 (0101)	Write Class: PMF (12) | TYPE (4) | TARID (8) | SRCID (8) | TTYPE (4) | WRSIZE (4) | SRCTID (8) ADDRESS (29) | W (1) | ADDRESS (2) | MSB for DW0 --> LSB for DWn TTYPE:0001 (FLUSH); 0010–0011 (Reserved); 0100 (NWRITE); 0101 (NWRITE_R); 0110–0111 (Reserved);
6 (0110)	Streaming-Write Class: PMF (12) | TYPE (4) | TARID (8) | SRCID (8) | ADDRESS (29) (1) | ADDRESS (2) | MSB for DW0 --> LSB for DWn

Table 16-83. RapidIO Packet Format Summary (tt = 0b00) (continued)

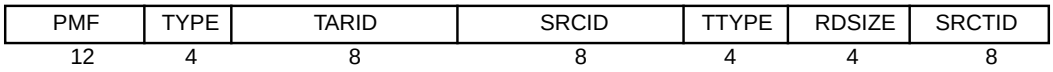
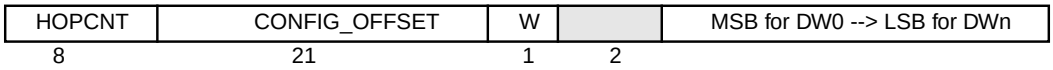
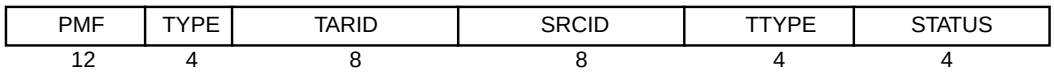
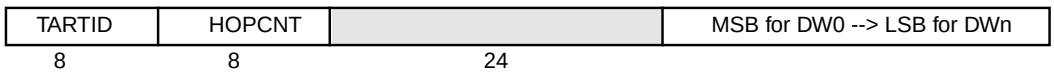
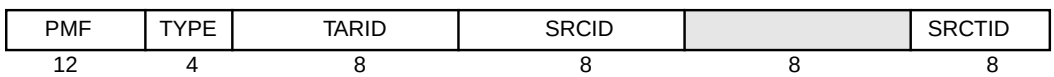
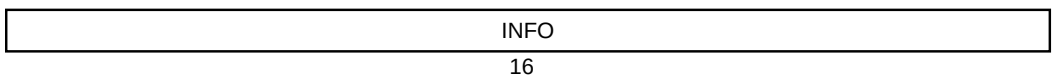
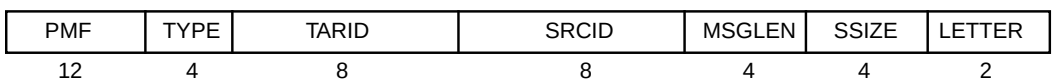
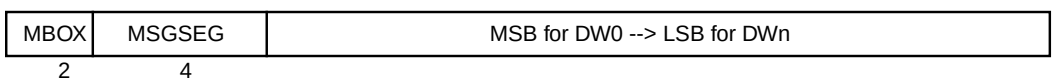
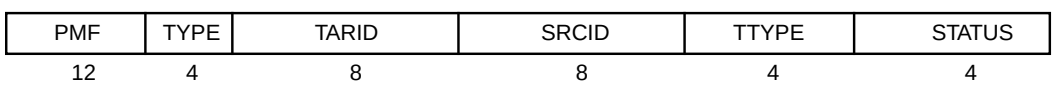
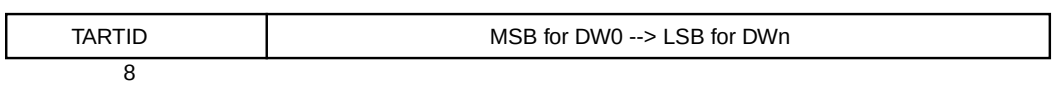
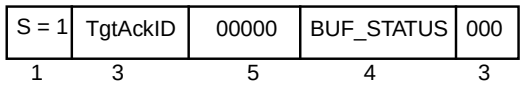
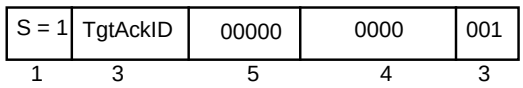
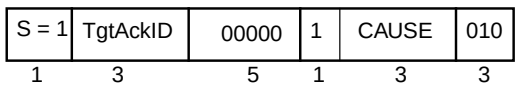
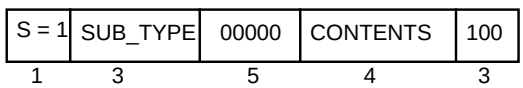
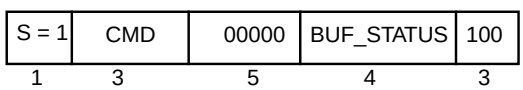
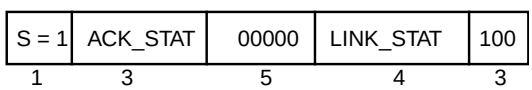
Type	Packet Format
8 (1000)	Maintenance Class Request:   TTYPE: 0000 (READ); 0001 (WRITE): 0100 (PORT WRITE) Maintenance Class Response:   TTYPE: 0010 (READ RESP); 0011 (WRITE RESP) STATUS: 0000 (DONE); 0111 (ERROR);
10 (1010)	Doorbell Class:  
11 (1011)	Message Class:  
13 (1101)	Response Class:   STATUS: 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 16-84 summarizes the defined control symbols, and RapidIO specific details are noted in the comments section of the table.

Table 16-84. Control Symbol Formats

Symbol Type	Symbol Format	Comments
0 (000)	Packet accepted: 	MPC8560 sets buf_status to 0xf when it generates a packet accepted control symbol.
1 (001)	Packet retry: 	None
2 (010)	Packet not accepted: 	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
4 (100)	Packet control: 	Sub_type (contents) 000: Idle (MPC8560 sets to 1111) 001: Stomp (unused) 010: EOP (MPC8560 sets to 1111) 011: Restart From Retry (unused) 100: Throttle (number of aligned pacing idles) 101: TOD-sync (number of max sized packets)
5 (101)	Link request: 	MPC8560 sets buf_status to 0xf when it generates a link request Cmd 000 Send training 010 Override width 011 Reset 100 Input status
6 (110)	Link response: 	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

16.5 RapidIO Functionality

16.5.1 General Device Functionality Checklists

The general device functionality checklists are applicable to devices of all classes, including end point free switch devices.

16.5.1.1 General Specification Functionality List

The list shown in Table 16-85 specifies general requirements for a compliant device.

Table 16-85. 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 16-85. 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

16.5.1.2 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.

16.5.1.3 Basic Functionality List

Table 16-86 lists requirements for basic functionality for the device.

Table 16-86. 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 16-86. 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

16.5.1.4 General Device Link Maintenance List

Table 16-87 lists requirements for link maintenance functionality.

Table 16-87. 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

16.5.1.5 General Device Packet Transmission List

Table 16-88 lists requirements for the device while transmitting a packet (a general device is required to generate response packets).

Table 16-88. 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

16.5.1.6 General Device Packet Reception List

Table 16-89 lists the requirements for the device to receive a packet.

Table 16-89. 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 16-89. 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

16.5.1.7 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 16-90 lists recoverable errors at the physical layer.

Table 16-90. 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 16-90. 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 16-90. 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

16.5.1.8 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 16-91. 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

16.5.2 Common Transport Layer Functionality Lists

The common transport layer functionality is broken down into a number of sub-lists.

16.5.2.1 Common Transport Layer Basic Functionality List

Table 16-92 lists the basic functionality for the device at the common transport layer.

Table 16-92. 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

16.5.2.2 Common Transport Layer Packet Transmission List

Table 16-93 lists the functionality for the device to transmit a packet (a slave device is required to generate response packets).

Table 16-93. 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

16.5.2.3 Common Transport Layer Packet Reception List

Table 16-94 lists the functionality for the device to receive a packet.

Table 16-94. 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

16.5.2.4 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 16-95. General Device Common Transport Layer Detectable Errors List

Item No.	Item Functionality	Meets Reqs?
1	Received reserved tt field encoding	Y

16.5.3 Logical Layer Functionality Lists

The logical layer functionality is broken down into a number of sub-lists.

16.5.3.1 Logical Layer Basic Functionality List

Table 16-96 lists the requirements for basic device functionality. A class 1 device is always the target of a request transaction.

Table 16-96. 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 16-96. 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 lsbs 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

16.5.3.2 Logical Layer Target Transaction Support List

Table 16-97 lists the requirements for the device to function as a transaction target.

Table 16-97. 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

16.5.3.3 Logical Layer Source Transaction Support List

General functionality does not require a device to initiate any operations.

16.5.3.4 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 16-98. 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

16.5.3.5 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.

16.6 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 16-99 lists all the errors detected by the RapidIO implementation and the actions taken.

Table 16-99. 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. MPC8560 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 16-99. 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 16-99. 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 16-99. 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 16-99. 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.

16.7 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 MPC8560 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

16.7.1 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.

16.7.1.1 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”.

16.7.1.2 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 meed 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.

16.7.2 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.

16.7.2.1 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.

16.7.2.2 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 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 16.8.4.1, “Data Message Controller,” Section 16.8.4.4, “Doorbell Message Controller,” and Section 16.8.4.5, “Port-Write Controller Structure.”

16.7.2.3 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 16.3.3.3, “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 interupt 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.

16.8 RapidIO Message Unit

16.8.1 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.

16.8.2 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

16.8.3 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.

16.8.4 RapidIO Messaging Description

This section relates to controller operations, formats, interrupts, and response conditions.

16.8.4.1 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.

16.8.4.2 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.

16.8.4.2.1 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.

16.8.4.2.2 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 16-80 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.

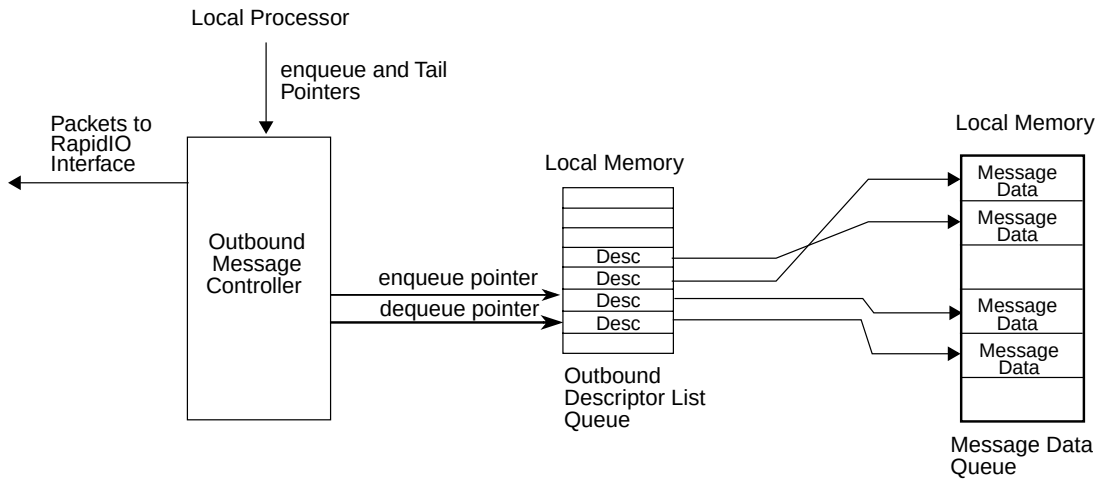


Figure 16-80. Outbound Message Queue Structure

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.

16.8.4.2.3 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.

16.8.4.2.4 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 16-100. 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 16-81 below depicts the queue dequeue pointer and an associated descriptor. The descriptor is only valid if the enqueue and dequeue pointers are not equal.

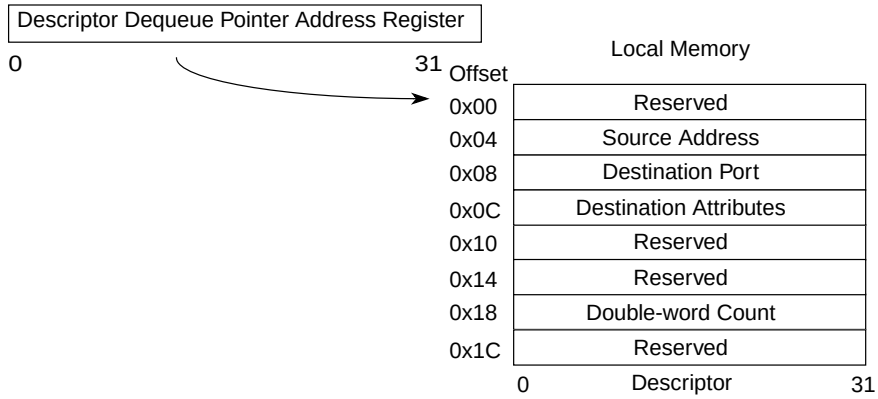


Figure 16-81. Descriptor Dequeue Pointer and Descriptor

16.8.4.2.5 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.

16.8.4.2.6 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.

16.8.4.3 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 16-82 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.

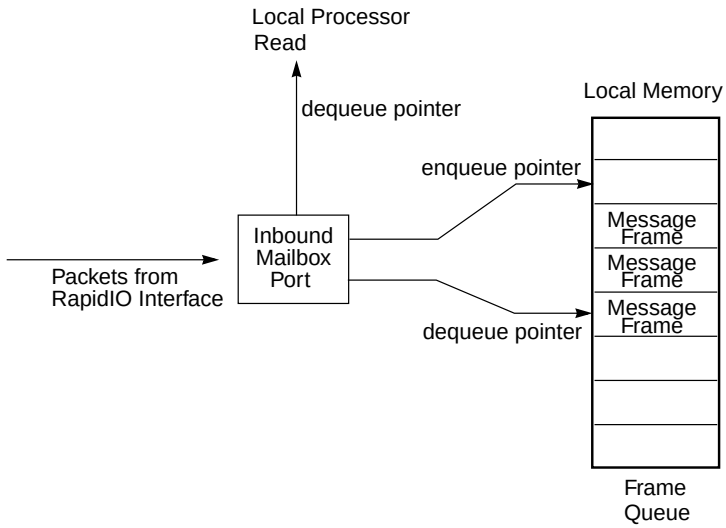


Figure 16-82. Inbound Mailbox Structure

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.

16.8.4.3.1 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.

16.8.4.3.2 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 (msgseg > 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.

16.8.4.3.3 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. ISR[MIQI] is set if IMR[MIQIE] is set.
- Queue full interrupt. This interrupt is generated each time the circular queue becomes full and the interrupt is enabled. ISR[QFI] is set if IMR[QFIE] is set.

16.8.4.3.4 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.

16.8.4.4 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 16-83 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.

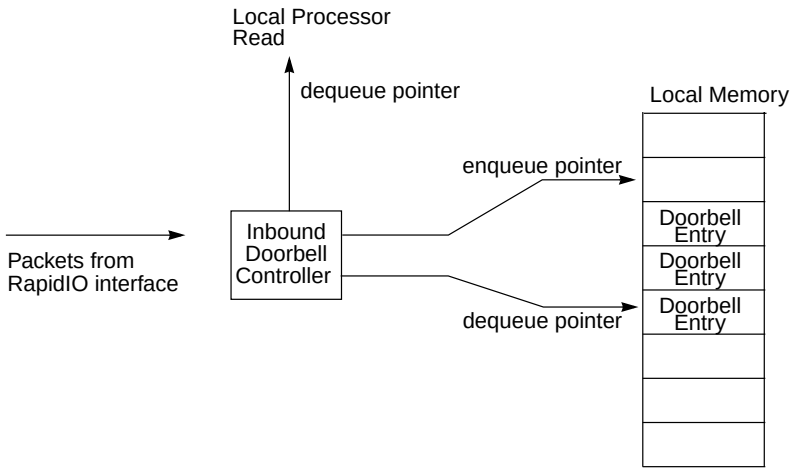


Figure 16-83. Inbound Doorbell Queue and Pointer Structure

16.8.4.4.1 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 (DMR[DE] = 1) and it is not busy with another doorbell message (DSR[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 (DMR[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.

16.8.4.4.2 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 16-101. Target Info Definition

Bits	Name	Description
0–23	—	Reserved
24–31	TID	Target ID field from the received doorbell packet

Table 16-102. 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 16-84 depicts the doorbell queue entry fields and their related offsets.

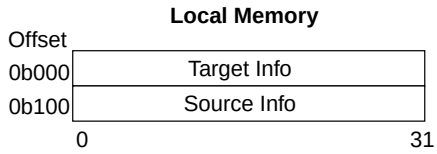


Figure 16-84. Doorbell Entry Format

16.8.4.4.3 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.

16.8.4.4.4 Error Response Conditions

Several conditions cause a logical layer error response (response “error”).

- Doorbell controller is error state.
- Doorbell controller is not enabled.

16.8.4.4.5 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.

16.8.4.5 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 16-85 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.

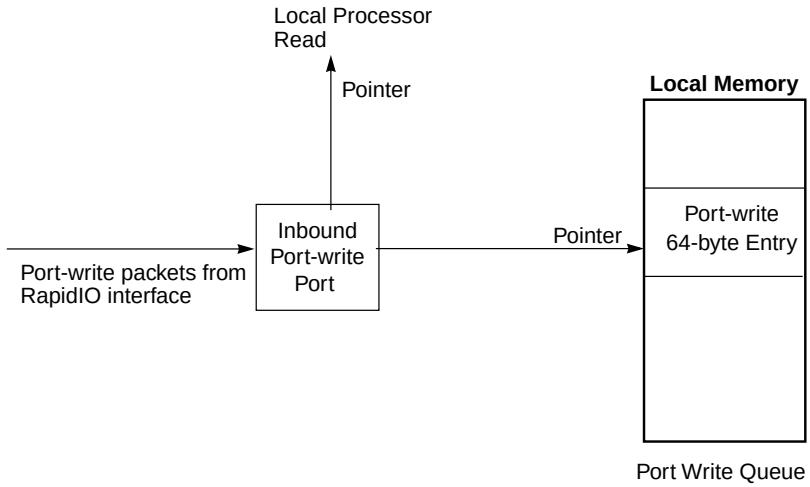


Figure 16-85. Inbound Port-Write Structure

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.

16.9 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 MPC8560. It contains the following chapters:

- Chapter 17, “Global Utilities,” defines the global utilities of the MPC8560. 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 18, “Performance Monitor,” describes the performance monitor of the MPC8560.
- Chapter 19, “Debug Features and Watchpoint Facility,” describes the debug features and watchpoint monitor of the MPC8560.

Chapter 17
Global Utilities

17.1 Introduction

This chapter describes the global utilities of the MPC8560. It provides signal descriptions, register descriptions and a functional description of these utilities.

17.1.1 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.

17.1.2 Global Utilities Features

This section provides a summary overview of the features that comprise the global utilities.

17.1.2.1 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)

17.1.2.2 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.

17.1.2.3 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.

17.1.2.4 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]

The multiplexing of the CPM signals occurs through the CPM programming model. See Chapter 42, “Parallel I/O Ports,” for details on CPM signal multiplexing.

17.1.2.5 Clock Control

The global utilities block also selects the internal clock signal driven on CLK_OUT.

17.2 External Signal Description

17.2.1 Signals Overview

Table 17-1 summarizes the external signals used by the global utilities block.

Table 17-1. 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 17-2
$\overline{\text{CKSTP_IN}}$	I	HIGH	Checkstop input	Digitally filtered	Table 17-2
$\overline{\text{CKSTP_OUT}}$	O	HIGH	Checkstop output	Negated with the negation of $\overline{\text{HRESET}}$	Table 17-2
CLK_OUT	O		Clock out. Selected by CLKOCR values.	—	Table 17-2
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

Table 17-1. External Signal Summary (continued)

Signal Name	I/O	Idle State	State Meaning	Timing/Comments	Section No.
$\overline{\text{SRESET}}$	I	HIGH	Soft reset	Digitally filtered	Chapter 4, “Reset, Clocking, and Initialization
$\overline{\text{HRESET}}$	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”

17.2.2 Detailed Signal Descriptions

Table 17-2 shows the detailed descriptions of the signals in the global utilities block.

Table 17-2. Detailed Signal Descriptions

Signal	I/O	Description	
$\overline{\text{CKSTP_IN}}$	I	Checkstop in	
		State Meaning	Asserted—Indicates that the MPC8560 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 17-2. Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{CKSTP_OUT}}$	O	Checkstop out	
		State Meaning	Asserted—Indicates that the e500 core of the MPC8560 is in a checkstop state. The system logic in the MPC8560 platform will remain functional . Negated—Indicates normal operation. After $\overline{\text{CKSTP_OUT}}$ has been asserted, it is negated after the next negation of $\overline{\text{HRESET}}$.
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks. Negation—Must remain asserted until the system has been reset with a hard reset.
ASLEEP	O	Asleep. See Section 17.4.1.6, “Sleep Mode.” After the negation of $\overline{\text{HRESET}}$ 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 17.3.1.16, “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].

17.3 Memory Map/Register Definition

Table 17-3 summarizes the global utilities registers and their addresses.

Table 17-3. 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	17.3.1.1/17-5
0xE_0004	POR boot mode status register(PORBMSR)	R	0xnnnn_0000	17.3.1.2/17-6
0xE_0008	POR I/O impedance status and control register (PORIMPSCR)	R/W	0x000n_003F	17.3.1.3/17-8
0xE_000C	POR I/O device status register (PORDEVSR)	R		17.3.1.4/17-9
0xE_0010	POR debug mode status register (PORDBGMSR)	R		17.3.1.5/17-10
0xE_0020	General-purpose POR configuration register (GPPORCR)	R		17.3.1.6/17-11
Signal Multiplexing and GPIO Controls				
0xE_0030	GPIO control register (GPIOCR)	R/W	0x0000_0000	17.3.1.7/17-11
0xE_0040	General-purpose output data register (GPOUTDR)	R/W	0x0000_0000	17.3.1.8/17-12
0xE_0050	General-purpose input data register (GPINDR)	R	0xnnnn_nnnn	17.3.1.9/17-13

Table 17-3. Global Utilities Block Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0xE_0060	Alternate function signal multiplex control (PMUXCR)	R/W	0x0000_0000	17.3.1.9/17-13
Device Disables				
0xE_0070	Device disable control (DEVDISR)	R/W	0x0000_0000	17.3.1.11/17-15
Power Management Registers				
0xE_0080	Power management status and control register (POWMGTCSR)	R/W	0x0000_0000	17.3.1.12/17-17
Interrupt Reporting				
0xE_0090	Machine check summary register (MCPSUMR)	Read/Clear	0x0000_0000	17.3.1.13/17-19
Version Registers				
0xE_00A0	Processor version register (PVR)	R	e500 processor version	17.3.1.14/17-20
0xE_00A4	System version register (SVR)	R	MPC8560 system version	17.3.1.15/17-20
Debug Control				
0xE_0E00	Clock out select register (CLKOCR)	R/W	0x0000_0000	17.3.1.16/17-21
0xE_0E10	DDR DLL control register (DDRDLLCR)	R/W	0x0000_0000	17.3.1.17/17-21
0xE_0E20	LBC DLL control register (LBDLLCR)	R/W	0x0000_0000	17.3.1.18/17-22

17.3.1 Register Descriptions

This section describes the global utilities registers in detail.

17.3.1.1 POR PLL Status Register (PORPLLSR)

Shown in Figure 17-1, the PORPLLSR contains the current settings for the PLL ratios as set by power-on reset (POR) configuration pins.

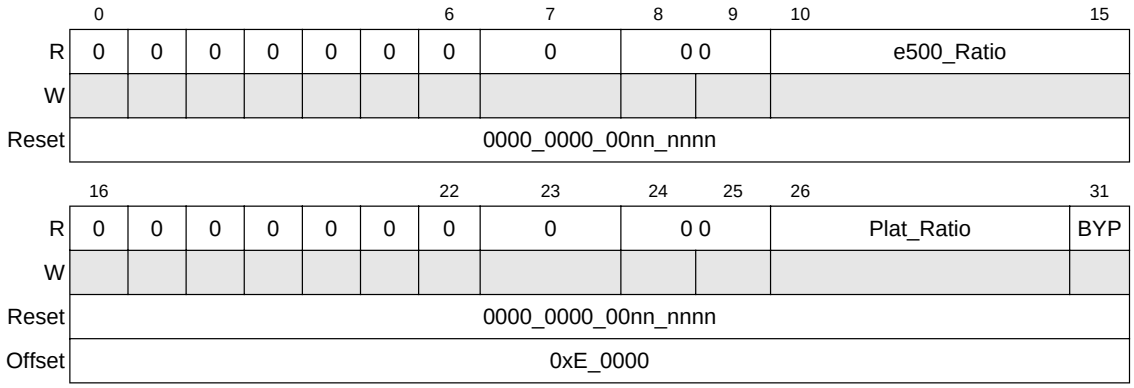


Figure 17-1. POR PLL Status Register (PORPLLSR)

Table 17-4 describes the bit settings of PORPLLSR.

Table 17-4. 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

17.3.1.2 POR Boot Mode Status Register (PORBMSR)

The PORBMSR, shown in Figure 17-2, 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 futher 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 16.3.2.19,

"Port General Control Command and Status Register (PGCCSR)." For more information about the PCI configurations, see Section 15.3.2.15, "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	0
W																
Reset	0000_0000_0000_0000															
Offset	0xE_0004															

Figure 17-2. POR Boot Mode Status Register (PORBMSR)

Figure 17-5 describes the bit settings of the PORBMSR.

Table 17-5. 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^2C addressing 10 Boot sequencer enabled with extended I^2C addressing 11 Boot sequencer disabled
12–13	—	Reserved
14–15	HA	Host/agent mode configuration. When the MPC8560 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

17.3.1.3 POR I/O Impedance Status and Control Register (PORIMPSCR)

Shown in Figure 17-3, 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 *MPC8560 Hardware Specification* for exact I/O impedences.

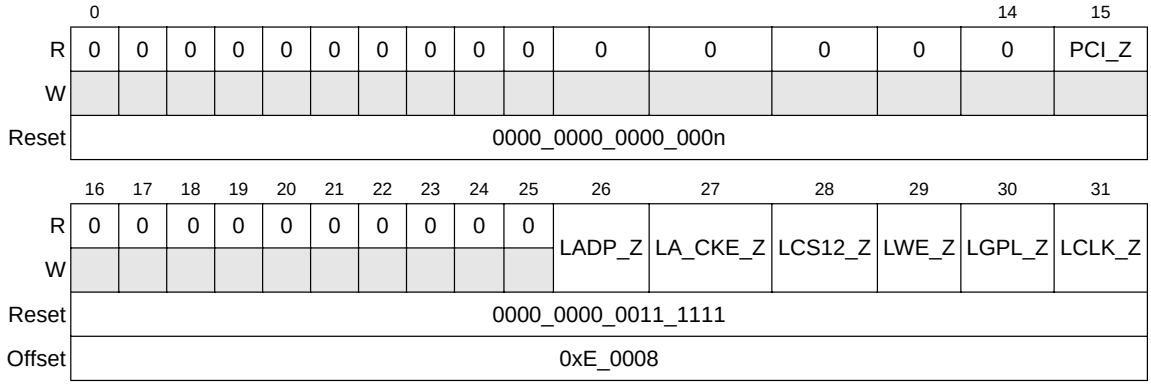


Figure 17-3. POR I/O Impedance Status and Control Register (PORIMPSCR)

Table 17-6 describes the bit settings of PORIMPSCR.

Table 17-6. PORIMPSCR Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	PCI_Z	PCI/PCI-X I/O impedance 0 Low impedence 1 High impedence
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 impedence 1 High impedence
27	LA_CKE_Z	I/O impedance for local bus address and clock enable (LA[27:31] and LCKE) 0 Low impedence 1 High impedence
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 impedence 0 Low impedence 1 High impedence
29	LWE_Z	I/O impedance for local bus write enables (LWE[0:3]) 0 Low impedence 1 High impedence

Table 17-6. PORIMPSCR Field Descriptions (continued)

Bits	Name	Description
30	LGPL_Z	I/O impedance for local bus general-purpose lines (LGPL0 - LGPL5) 0 Low impedence 1 High impedence
31	LCLK_Z	I/O impedance for local bus clocks (LCLK[0:2]) 0 Low impedence 1 High impedence

17.3.1.4 POR Device Status Register (PORDEVSR)

Shown in Figure 17-4, 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 17-4. POR Device Status Register (PORDEVSR)

Table 17-7 describes the bit settings of PORDEVSR.

Table 17-7. 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 17-7. 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

17.3.1.5 POR Debug Mode Status Register (PORDBGMSR)

Shown in Figure 17-5, the PORDBGMSR contains the debug mode settings from the power-on-reset configuration pins.

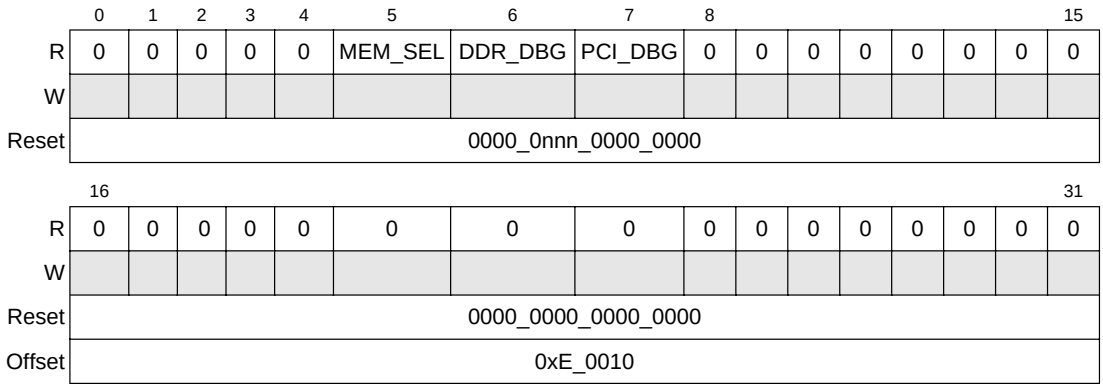


Figure 17-5. POR Debug Mode Status Register (PORDBGMSR)

Table 17-8 describes the bit settings of PORDBGMSR.

Table 17-8. 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 17-8. 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

17.3.1.6 General-Purpose POR Configuration Register (GPPORCR)

Shown in Figure 17-6, the GPPORCR contains the genera-purpose power-on-reset configuration register value sampled from the local bus address/data signals LAD[0:31].

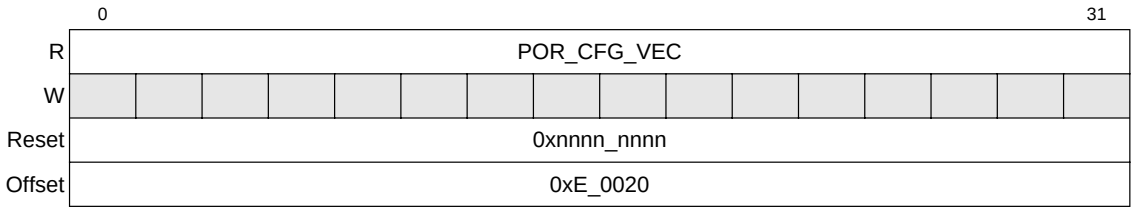


Figure 17-6. POR Configuration Register (GPPORCR)

Table 17-9 describes the bit settings of GPPORCR.

Table 17-9. 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 $\overline{\text{HRESET}}$.

17.3.1.7 General-Purpose I/O Control Register (GPIOCR)

Shown in Figure 17-7, 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.

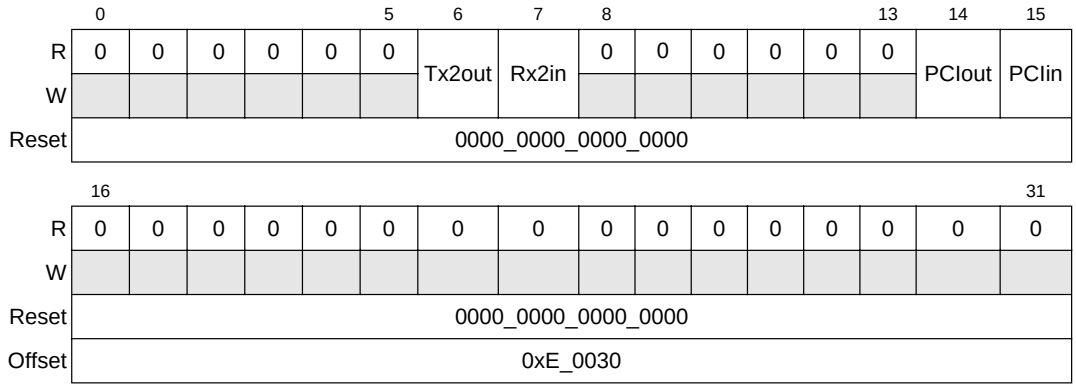


Figure 17-7. General-Purpose I/O Control Register (GPIOCR)

Table 17-10 describes the bit settings of GPIOCR.

Table 17-10. 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	PCIout	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	PCIin	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

17.3.1.8 General-Purpose Output Data Register (GPOUTDR)

Shown in Figure 17-8, 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.

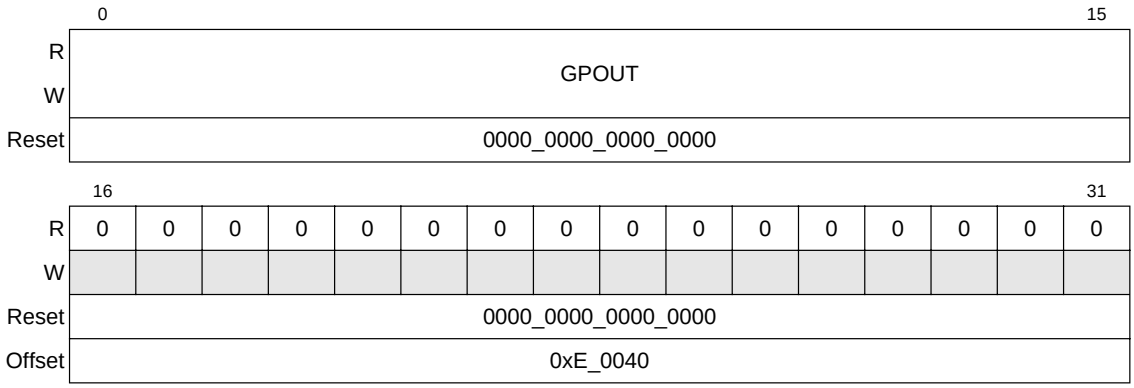


Figure 17-8. General-Purpose Output Data Register (GPOUTDR)

Table 17-11 describes the fields of GPOUTDR.

Table 17-11. 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

17.3.1.9 General-Purpose Input Data Register (GPINDR)

Shown in Figure 17-9, 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.

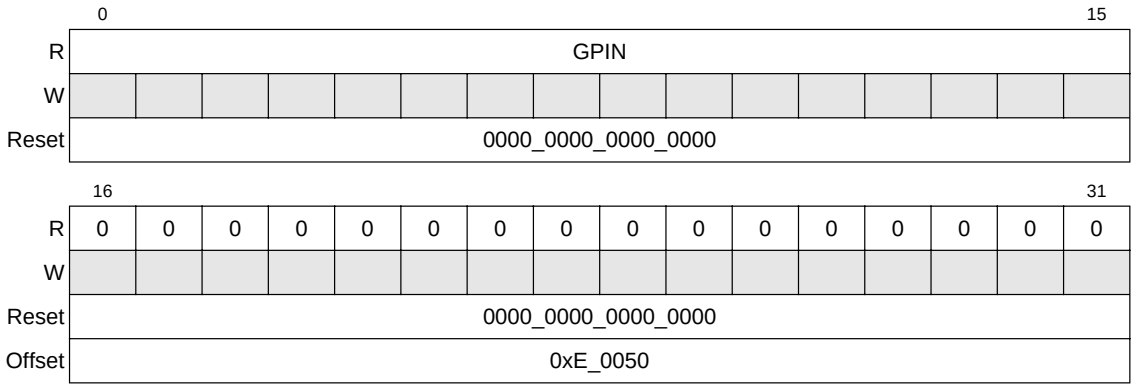


Figure 17-9. General-Purpose Output Data Register (GPINDR)

Table 17-11 describes the fields of GPINDR.

Table 17-12. 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

17.3.1.10 Alternate Function Signal Multiplex Control Register (PMUXCR)

Shown in Figure 17-10, 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. Note that CPM signal multiplexing is handled separately through the CPM programming model described in Chapter 42, “Parallel I/O Ports.”

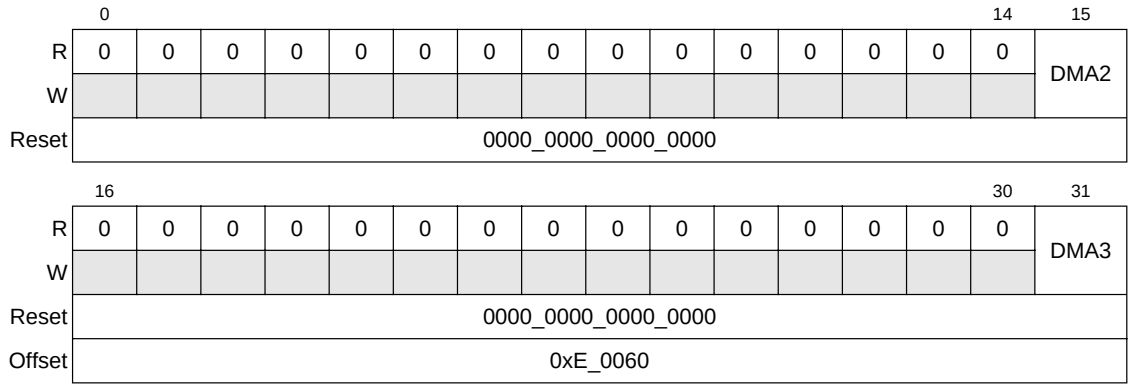


Figure 17-10. Alternate Function Pin Multiplex Control Register (PMUXCR)

Table 17-13 describes the bit settings of PMUXCR.

Table 17-13. 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 $\overline{\text{DMA_DREQ2}}$ LCS6 acts as $\overline{\text{DMA_DACK2}}$ LCS7 acts as $\overline{\text{DMA_DDONE2}}$
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 $\overline{\text{DMA_DREQ3}}$ IRQ10 acts as $\overline{\text{DMA_DACK3}}$ IRQ11 acts as $\overline{\text{DMA_DDONE3}}$

17.3.1.11 Device Disable Register (DEVDISR)

Shown in Figure 17-11, 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.

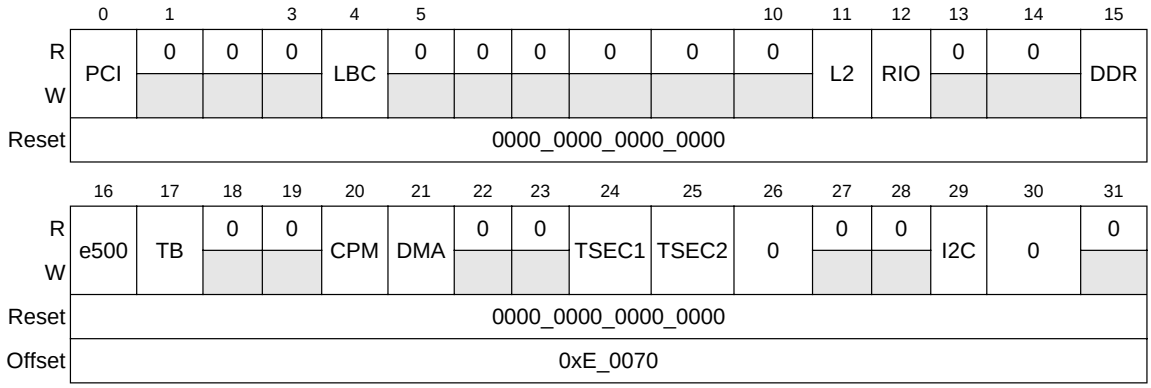


Figure 17-11. Device Disable Register (DEVDISR)

Table 17-14 describes the bit settings of DEVDISR

Table 17-14. 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.
18–19	—	Reserved

Table 17-14. DEVDISR Field Descriptions (continued)

Bits	Name	Description
20	CPM	Communications processor module disable 0 CPM enabled 1 CPM disabled
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. RxD and TxD pins may be used for general-purpose I/O
26–28	—	Reserved
29	I2C	I^2C controller disabled 0 I^2C controller enabled 1 I^2C controller disabled
30–31	—	Reserved

17.3.1.12 Power Management Control and Status Register (POWMGTCSR)

Shown in Figure 17-12, the POWMGTCSR contains control bits for placing the MPC8560 into low power states and for controlling when it wakes up. POWMGTCSR also contains status bits for reporting the current power management state.

	0	1	2				6	7	8			11	12	13	14	15
R	IRQ_MSK	CI_MSK	0	0	0	0	0	QPCI_ MASK	0	0	0	0	DOZ	0	SLP	QSC
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	DOZING	NAPPING	SLPING	QSCED
W																
Reset	0000_0000_0000_0000															
Offset	0xE_0080															

Figure 17-12. Power Management Control & Status Register (POWMGTCSR)

Table 17-15 describes the bit settings of POWMGTCSR.

Table 17-15. POWMGTCSR 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 quiece 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 POWMGTCSR[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 17-15. POWMGTCSR 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 POWMGTCSR[SLP] bit is set in the system or because the HID0[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 POWMGTCSR[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 POWMGTCSR[QPCI_MSK] bit is set) may master reads and writes into the system. All other mastering interfaces are shut down.

17.3.1.13 Machine Check Summary Register (MCPSUMR)

Shown in Figure 17-13, 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.

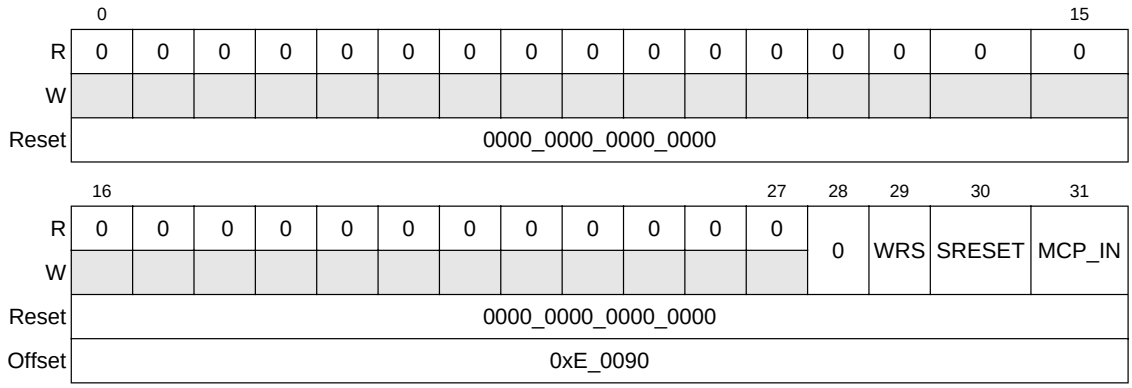


Figure 17-13. Machine Check Summary Register (MCPSUMR)

Table 17-16 describes the bit settings of MCPSUMR.

Table 17-16. 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 17-16. 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.

17.3.1.14 Processor Version Register (PVR)

Shown in Figure 17-14, 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).

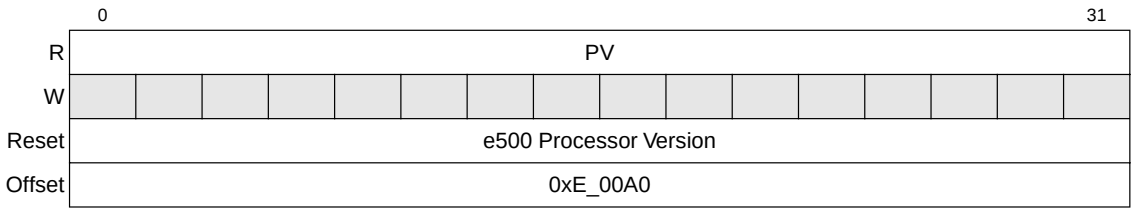


Figure 17-14. Processor Version Register (PVR)

Table 17-17 describes the fields of PVR.

Table 17-17. PVR Field Descriptions

Bits	Name	Description
0–31	PV	Processor version for the e500 CPU

17.3.1.15 System Version Register (SVR)

Shown in Figure 17-15, the SVR contains the system version number for the MPC8560 system logic. Note that value in this memory-mapped register is copied in the SVR register of the e500 core.

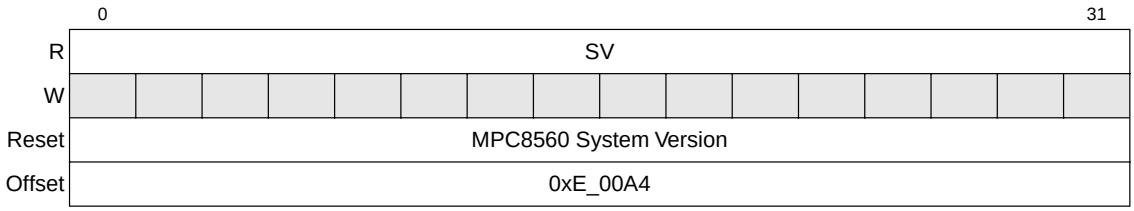


Figure 17-15. System Version Register (SVR)

Table 17-18 describes the fields of SVR.

Table 17-18. SVR Field Descriptions

Bits	Name	Description
0–31	SV	System version for the MPC8560 system logic

17.3.1.16 Clock Out Control Register (CLKOCR)

Shown in Figure 17-16, the CLKOCR contains control bits that select the clock sources to be placed on the clock out (CLK_OUT) signal.

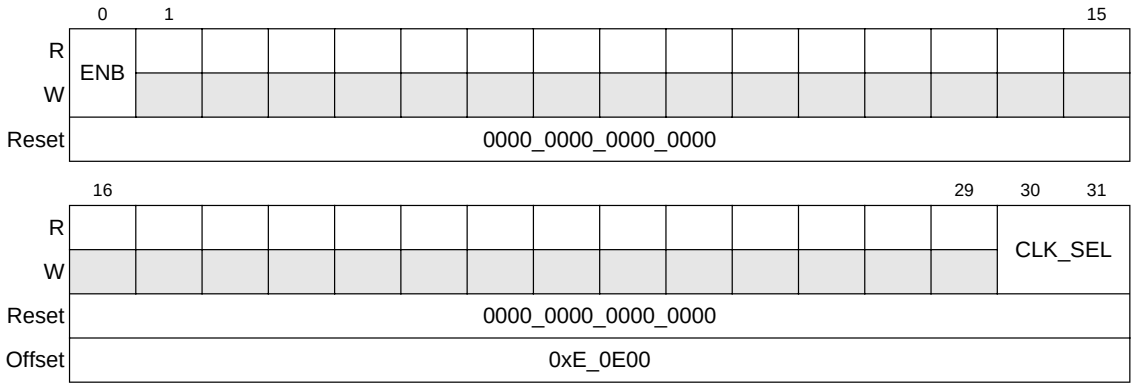


Figure 17-16. Clock Out Control Register (CLKOCR)

Table 17-19 describes the bit settings of CLKOCR.

Table 17-19. 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)

17.3.1.17 DDR DLL Control Register (DDRDLLCR)

Shown in Figure 17-17, 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.

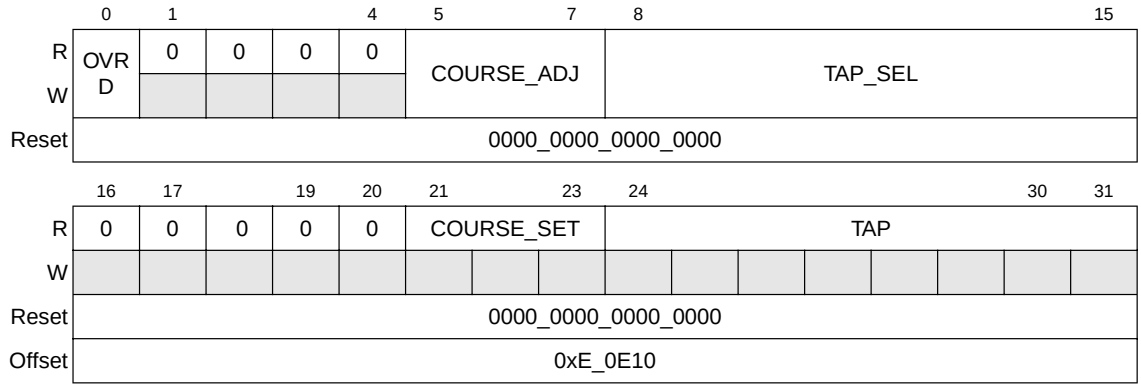


Figure 17-17. DDR DLL Control Register (DDRDLLCR)

Table 17-20 describes the bit settings of DDRDLLCR.

Table 17-20. 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.

17.3.1.18 Local Bus DLL Control Register (LBDLLCR)

Shown in Figure 17-18, 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.

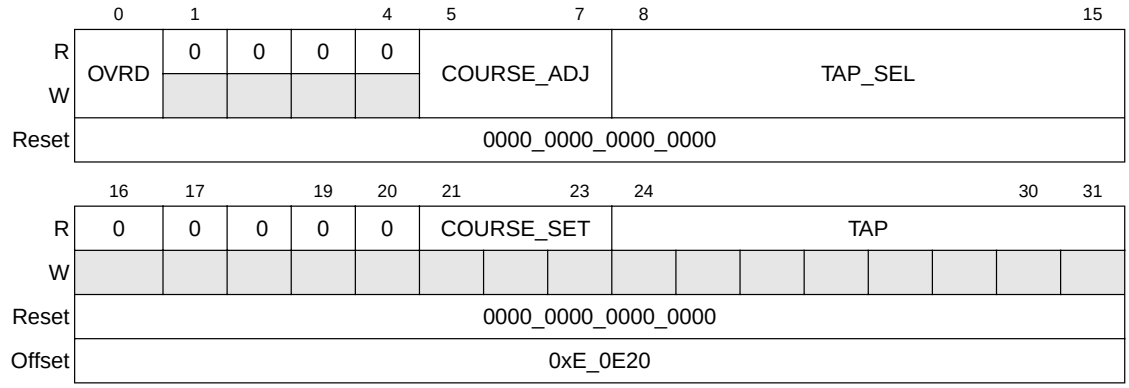


Figure 17-18. Local Bus DLL Control Register (LBDLLCR)

Table 17-21 describes the bit settings of LBDLLCR.

Table 17-21. 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.

17.4 Functional Description

This section describes the global utilities from a functional perspective.

17.4.1 Power Management

The MPC8560 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.

17.4.1.1 Dynamic Power Management

Many blocks in the MPC8560 are capable of dynamically turning off clocks within the block when sections of the block are idle.This power savings feature is always enabled.

17.4.1.2 Shutting Down Unused Blocks

Several blocks within the MPC8560 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 17.3.1.11, “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.

17.4.1.3 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 MPC8560 to cause the device to enter doze or sleep modes.

17.4.1.4 Doze Mode

In doze mode, the e500 processor core of the MPC8560 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 MPC8560 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 17.4.1.8, “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.

17.4.1.5 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 MPC8560 returns to nap mode after the interrupt is serviced. See Section 17.4.1.8, "Interrupts and Power Management," for more information.

System logic external to the e500 core remains fully operational in doze mode.

17.4.1.6 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 MPC8560 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 MPC8560 when it is asleep.

17.4.1.7 Power Management Control Bits

The e500 core provides the following control bits that are used to signal the power management requests to the MPC8560 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 MPC8560 to initiate doze mode.
- HID0[NAP]—Qualified with MSR[WE], signals the MPC8560 to initiate nap mode.
- HID0[SLEEP]—Qualified with MSR[WE], signals the MPC8560 to initiate sleep mode.

Refer to the *e500 User's Manual* for more details on accessing these power management control bits.

17.4.1.8 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 MPC8560 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.

17.4.1.9 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 MPC8560 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.

17.4.1.10 Software Considerations for Power Management

Setting MSR[WE] generates a request to the MPC8560 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
```

17.4.2 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 17.3.1.7, "General-Purpose I/O Control Register (GPIOCR)."

17.4.3 Interrupt and Local Bus Signal Multiplexing

Except for the CPM, the MPC8560 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 17.3.1.10, "Alternate Function Signal Multiplex Control Register (PMUXCR)."

The multiplexing of the CPM signals occurs through the CPM programming model. See Chapter 42, “Parallel I/O Ports,” for details on CPM signal multiplexing.

Chapter 18
Performance Monitor

18.1 Introduction

The MPC8560 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 MPC8560'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.

18.1.1 Overview

Figure 18-1 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.

- 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.

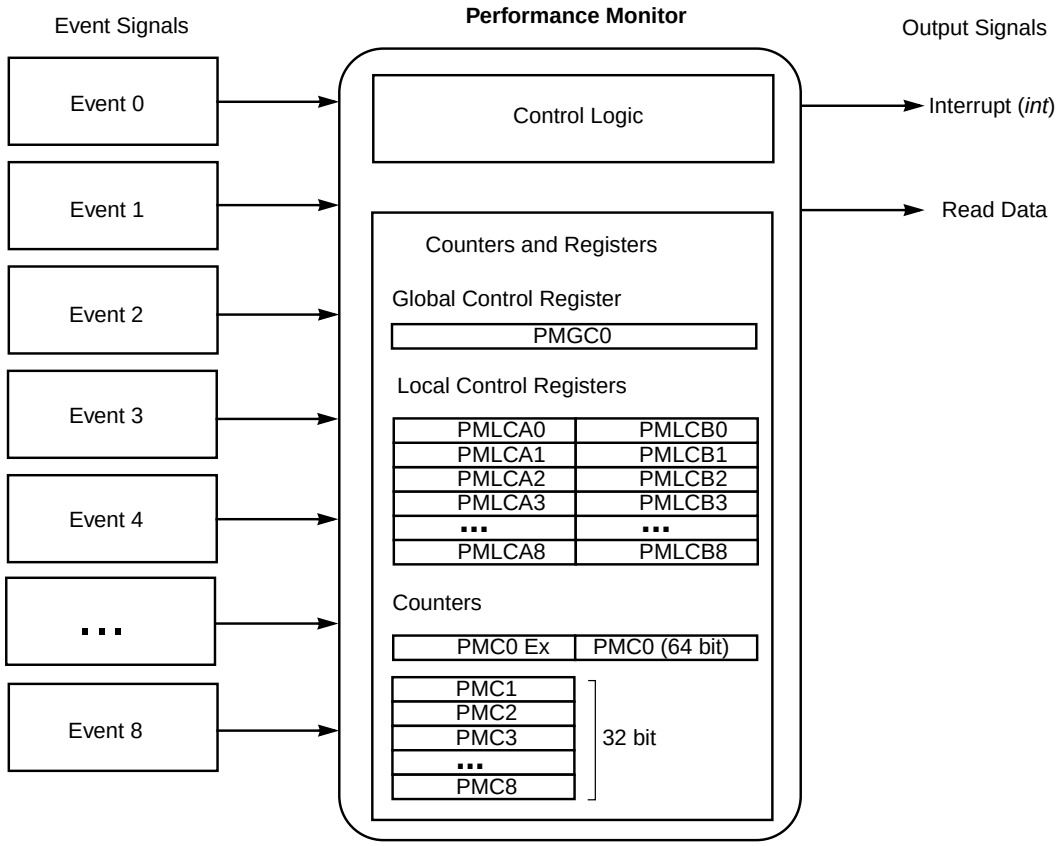


Figure 18-1. Performance Monitor Block Diagram

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.

18.1.2 Features

The MPC8560 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

18.2 External Signals

Table 18-1 outlines the signals of the MPC8560 performance monitor facility. The inputs to the MPC8560 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 18-1. Performance Monitor Interface Signals

Group	Signal	I/O	Comments
Performance Monitor Interrupt	*int*	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 18.4.7, “Performance Monitor Events”

18.3 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 18-2 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.

18.3.1 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 18-2. Control Register Memory Map

Address Offset (in Hex)	Register	Access	Reset	Section/Page
0xE_1000	Global control register (PMGC0)	R/W	0x0000_0000	18.3.2.1/18-5
0xE_1010	Local control register A0 (PMLCA0)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1014	Local control register B0 (PMLCB0)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1018	Performance monitor counter 0 (PMC0)	R/W	0x0000_0000	18.3.3.1/18-10
0xE_1010	Performance monitor counter 0 Extended (PMC0 Ex)	R/W	0x0000_0000	
0xE_1020	Local control register A1 (PMLCA1)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1024	Local control register B1 (PMLCB1)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1028	Performance monitor counter 1 (PMC1)	R/W	0x0000_0000	18.3.3.1/18-10
0xE_1030	Local control register A2 (PMLCA2)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1034	Local control register B 2 (PMLCB2)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1038	Performance monitor counter 2 (PMC2)	R/W	0x0000_0000	18.3.3.1/18-10
0xE_1040	Local control register A3 (PMLCA3)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1044	Local control register B3 (PMLCB3)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1048	Performance monitor counter 3 (PMC3)	R/W	0x0000_0000	18.3.3.1/18-10
0xE_1050	Local control register A4 (PMLCA4)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1054	Local control register B4 (PMLCB4)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1058	Performance monitor counter 4 (PMC4)	R/W	0x0000_0000	18.3.3.1/18-10
0xE_1060	Local control register A5 (PMLCA5)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1064	Local control register B 5 (PMLCB5)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1068	Performance monitor counter 5 (PMC5)	R/W	0x0000_0000	18.3.3.1/18-10
0xE_1070	Local control register A6 (PMLCA6)	R/W	0x0000_0000	18.3.3.1/18-10
0xE_1074	Local control register B6 (PMLCB6)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1078	Performance monitor counter 6 (PMC6)	R/W	0x0000_0000	18.3.3.1/18-10
0xE_1080	Local control register A7 (PMLCA7)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1084	Local control register B7 (PMLCB7)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1088	Performance monitor counter 7 (PMC7)	R/W	0x0000_0000	18.3.3.1/18-10
0xE_1090	Local control register A8 (PMLCA8)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1094	Local control register B8 (PMLCB5)	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1098	Performance monitor counter 8 (PMC8)	R/W	0x0000_0000	18.3.3.1/18-10

18.3.2 Control Registers

This section describes the performance monitor control registers in detail.

18.3.2.1 Performance Monitor Global Control Register (PMGC0)

The performance monitor global control register (PMGC0), shown in Figure 18-2, is a 32-bit register used to control all performance monitor counters.

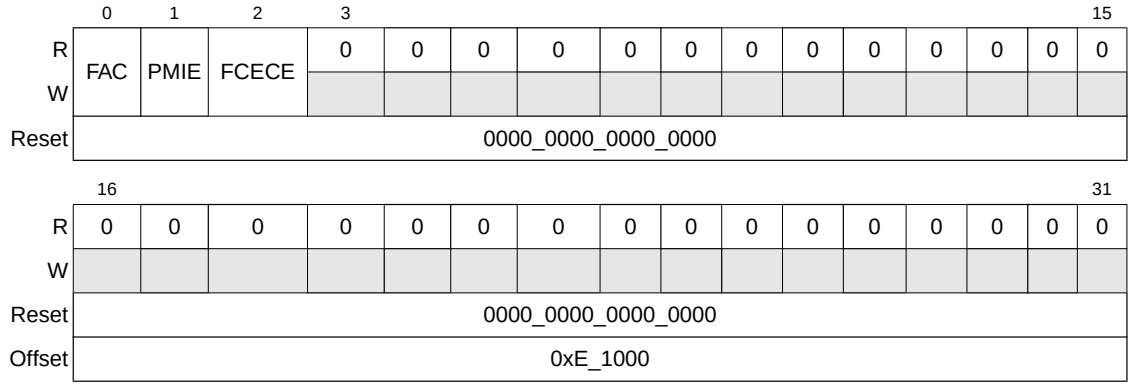


Figure 18-2. Performance Monitor Global Control Register (PMGC0)

Table 18-3 shows the field descriptions for the performance monitor global control register (PMGC0).

Table 18-3. 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 PMLCA*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

18.3.2.2 Performance Monitor Local Control Registers (PMLCA*n* and PMLCB*n*)

The performance monitor local control registers (PMLCA*n* and PMLCB*n*) are used to control the operation of the performance monitor counters. The performance monitor local

control A and B registers are paired 32-bit control registers that are associated with an individual counter to specify how the counter is used and what event is monitored on that counter. Figure 18-3 shows the performance monitor local control A0 register (PMLCA0).

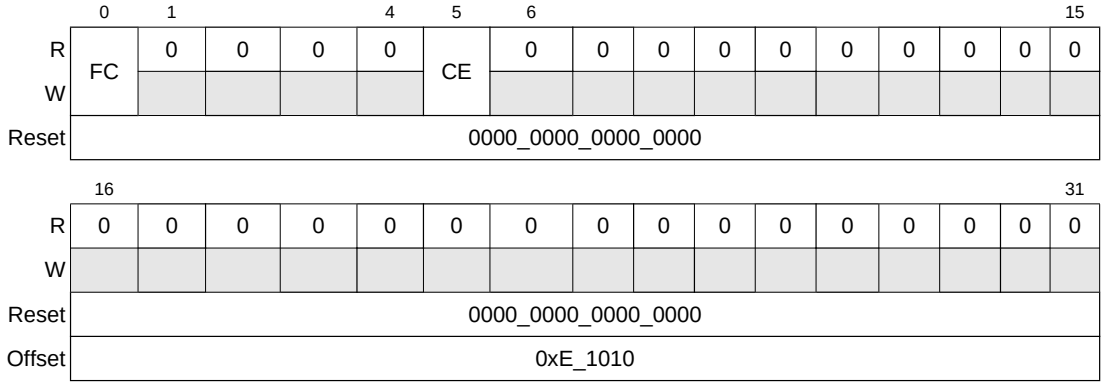


Figure 18-3. Performance Monitor Local Control Register A0 (PMLCA0)

Table 18-4 shows the field descriptions for the performance monitor local control A0 register (PMLCA0).

Table 18-4. 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 18-4 shows the performance monitor local control registers A1–A8.

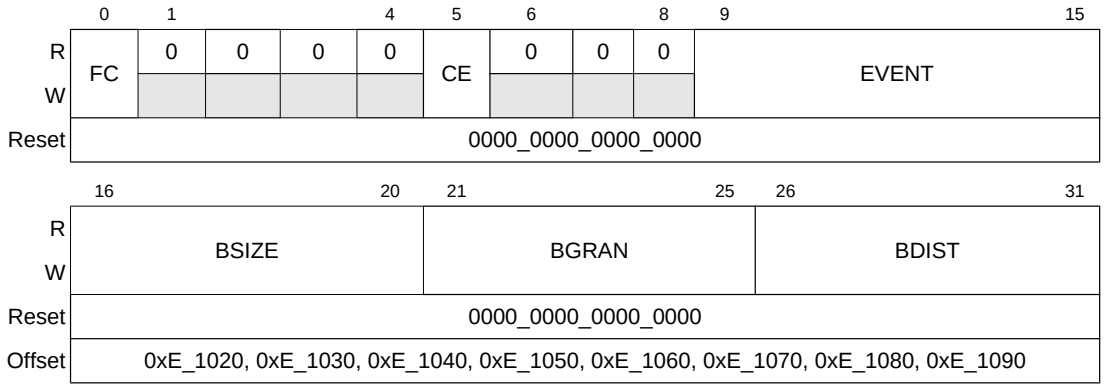


Figure 18-4. Performance Monitor Local Control Register A[1–8] (PMLCA[1–8])

Table 18-5 shows the field descriptions for the performance monitor local control A[1–8] registers (PMLCA*n*).

Table 18-5. 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 18.4.6, “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 18-5 shows the performance monitor local control B0 register (PMLCB0).

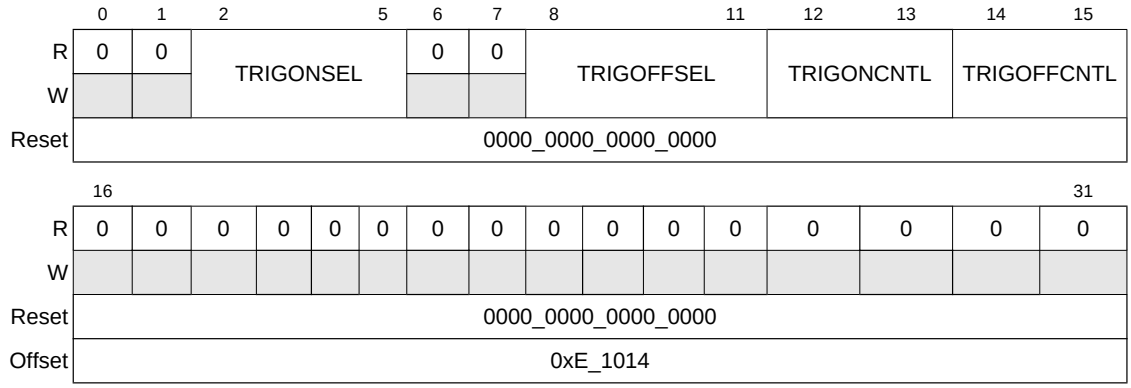


Figure 18-5. Performance Monitor Local Control Register B0 (PMLCB0)

Table 18-6 shows the field descriptions for the performance monitor local control B0 register (PMLCB0).

Table 18-6. 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 18-6 shows the performance monitor local control registers B1–B8.

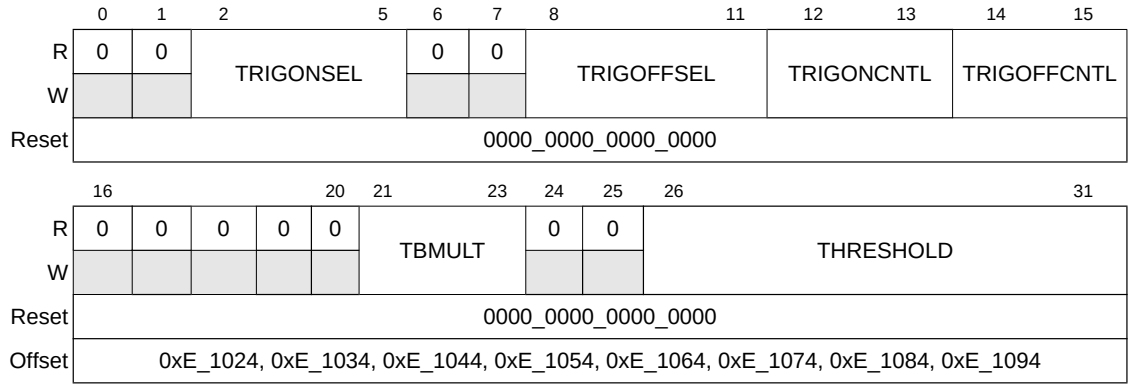


Figure 18-6. Performance Monitor Local Control Register B[1–8] (PMLCB[1–8])

Table 18-6 shows the field descriptions for the performance monitor local control B[1–8] registers (PMLCB*n*).

Table 18-7. 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 18-7. 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.

18.3.3 Counter Registers

This section describes the performance monitor counter registers in detail.

18.3.3.1 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 18-7 and Figure 18-8 show the two registers which comprise the 64-bit performance monitor counter. Figure 18-7 shows the upper 32-bits of the counter and Figure 18-8 shows the lower 32-bits.

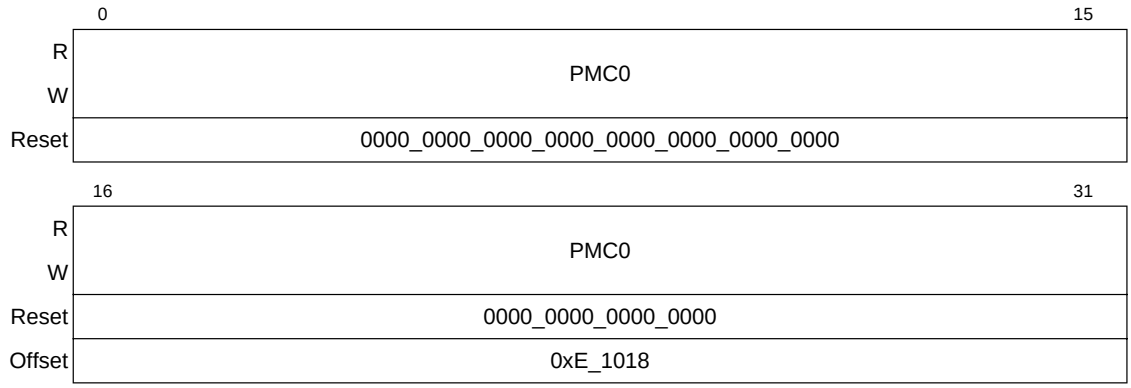


Figure 18-7. Performance Monitor Counter Register 0 Extended (PMC0) (Upper 32-bits)

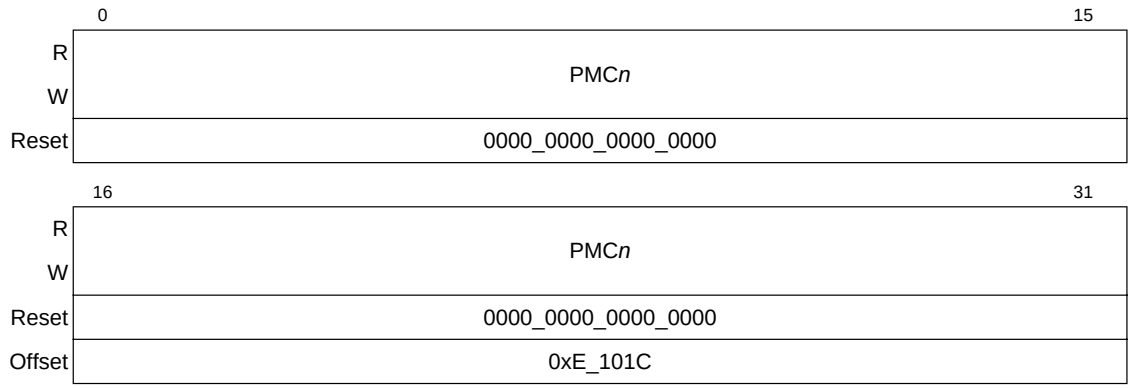


Figure 18-8. Performance Monitor Counter Register 0 (PMC0) (Lower 32-bits)

Table 18-8. 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 18-9 shows the performance monitor counters (PMC[1-8]).

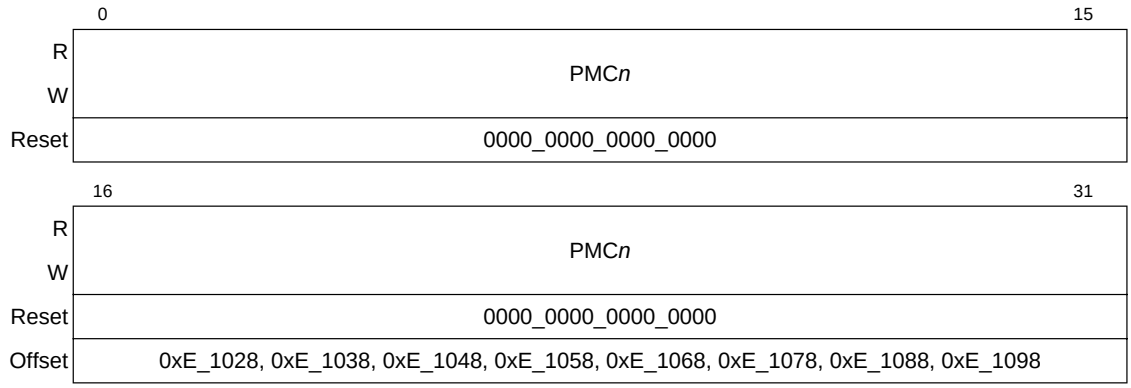


Figure 18-9. Performance Monitor Counter Register [1–8] (PMC[1–8])

Table 18-9 shows the field descriptions for the performance counters [1–8].

Table 18-9. PMC[1–8] Field Descriptions

Bits	Name	Description
0–31	PMC*n*	MSB indicates overflow. Setting the MSB can cause an immediate interrupt.

18.4 Functional Description

This section describes the use of some of the features of the performance monitor.

18.4.1 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 (PMLCA*n*[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.

18.4.2 Event Counting

Using the control registers described in Section 18.3.2, “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.

18.4.3 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 18-10.

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.

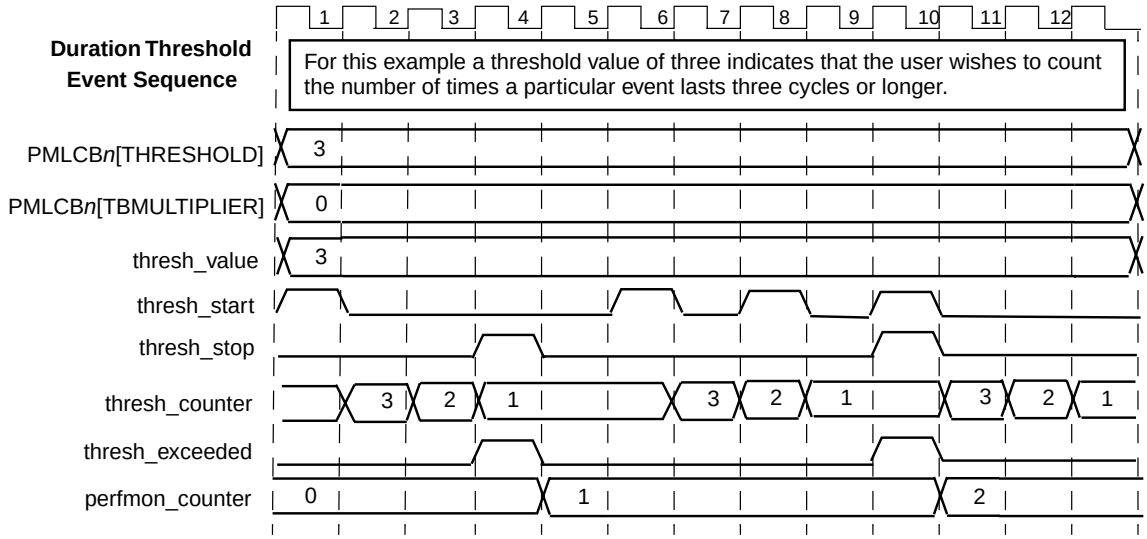


Figure 18-10. Duration Threshold Event Sequence Timing Diagram

18.4.4 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 18-11, 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, PMLCA*n*[CE] should be cleared to avoid signalling an interrupt when the counter's most significant bit is set. Note that several cycles may be required for the chained counters to reflect the true count because of the internal delay between when an overflow occurs and a counter increments.

18.4.5 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 (PMLCB*n*[TRIGONSEL]) or trigger off select (PMLCB*n*[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 (PMLCB*n*[TRIGONCNTL]) or trigger-off-control (PMLCB*n*[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.

18.4.6 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 18-10 event bursts are defined by three parameters; size, granularity, and distance. Figure 18-11 shows the same information graphically.

Table 18-10. Burst Definition

Parameter	Description	PMLCA*n* 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.

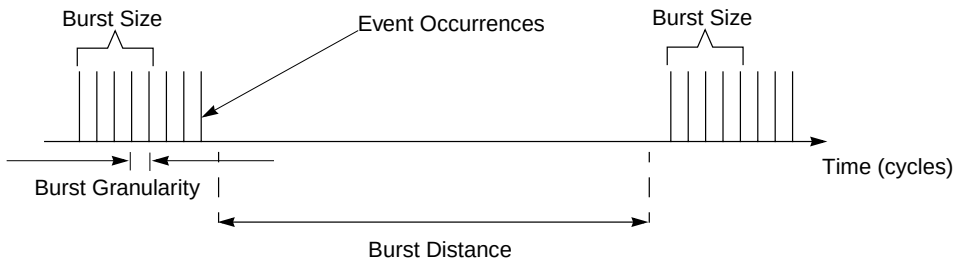


Figure 18-11. Burst Size, Distance, Granularity, and Burstiness Counting

The burstiness size field (PMLCA*n*[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 (PMLCA*n*[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 (PMLCA*n*[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 (PMLCA*n*[BDIST]) along with the threshold/burstiness multiplier field (PMLCB*n*[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 18-12 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.

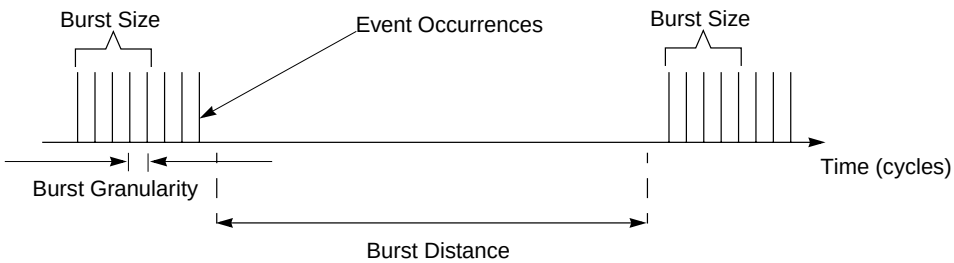


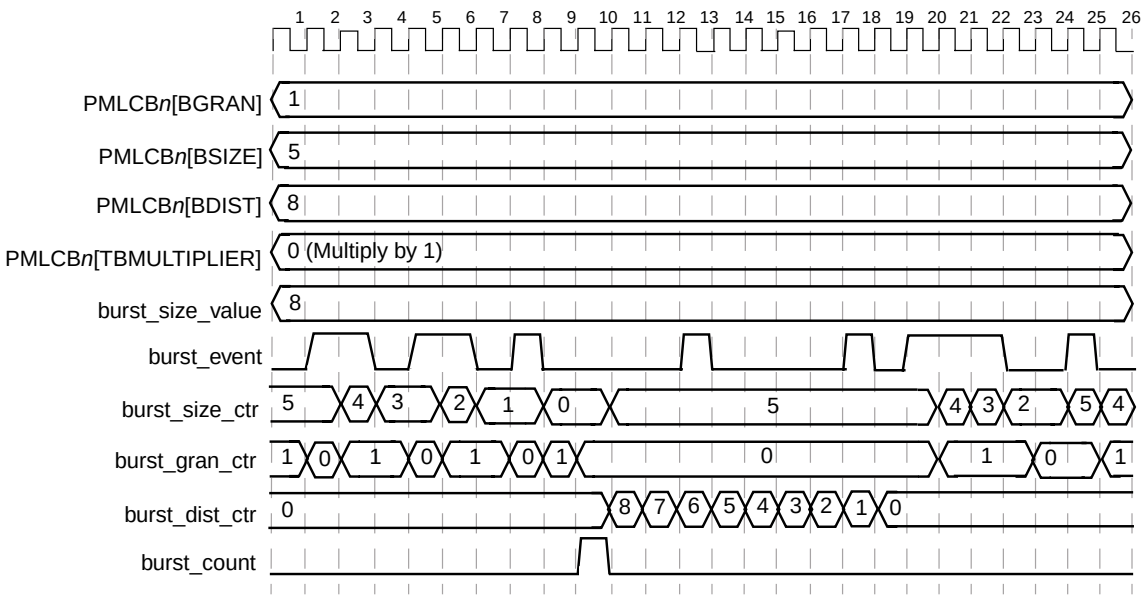
Figure 18-12. Description of Burstiness Counting

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 PMLCB*n*[BDIST] and PMLCB*n*[TBMULT] when a burst sequence has been identified and counted. This counter is decremented when burstiness counting is enabled and the counter has not already reached 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 18-13.



Note: For this example: count bursts of 5 event occurrences, burst granularity of 1, and acceptable distance between bursts of 8.

Figure 18-13. Burstiness Counting Timing Diagram

18.4.7 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

programmed to 64. Counter events that are not specified in Table 18-11 below are reserved.

In Table 18-11 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 18-11. 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 18-11. 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 CPM	C5:1	Total number of reads or writes from CPM
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 18-11. 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 CPM	C8:1	Total number of Row Open Table hits for reads or writes from CPM
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.

Table 18-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
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.
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 18-11. 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 18-11. 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 18-11. 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 18-11. 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 18-11. 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 18-11. 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 CPM	C7:13	Asserted for every cycle CPM 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 CPM	C2:20	ECM dispatch from CPM
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

Table 18-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
ECM dispatch from DMA	C7:14	ECM dispatch from DMA
ECM dispatch from other	C8:14	ECM dispatch from other
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 CPM	C2:23	ECM Data Bus grant CPM
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

Table 18-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
ECM Data Bus wait CPM	C6:21	ECM Data Bus CPM 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
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

Table 18-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
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		
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-bitoutbound 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

Table 18-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
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
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

Table 18-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
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
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		

Table 18-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
DMA reads	C1:47	Number of reads (descriptor and data)
Buffer descriptor reads	C2:54	Number of transmit and receive buffer descriptor reads
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.

Table 18-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
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
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 18-11. 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 18-11. 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 CPM port	C1:52	Number of requests granted to CPM port
Number of requests granted to ECM port	C2:57	Number of requests granted to ECM port
Number of cycles atomic lock for CPM port is enabled	C3:56	Number of cycles atomic lock for CPM port is enabled
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 CPM port	C5:49	Number of atomic lock time-outs for CPM port
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

Table 18-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
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
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
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.

Table 18-11. Performance Monitor Events (continued)

Event	Event Assignment	Description
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.

18.4.8 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 18-12 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 18-12. Register Settings for Counting Examples MPC8560

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 19
Debug Features and Watchpoint Facility

19.1 Introduction

This document describes all customer visible debug modes of the MPC8560 integrated device. The debug features on the MPC8560 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 MPC8560 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 19.4.1, “PCI/PCI-X Interface Debug”)
- DDR SDRAM interface debug (Section 19.4.2, “DDR SDRAM Interface Debug”)
- Local bus interface debug (Section 19.4.3, “Local Bus Interface Debug”)
- Watchpoint monitor and trace buffer debug (Section 19.4.4, “Watchpoint Monitor” and Section 19.4.5, “Trace Buffer”)

Figure 19-1 is a block diagram that shows the details pertinent to the customer-visible debug features.

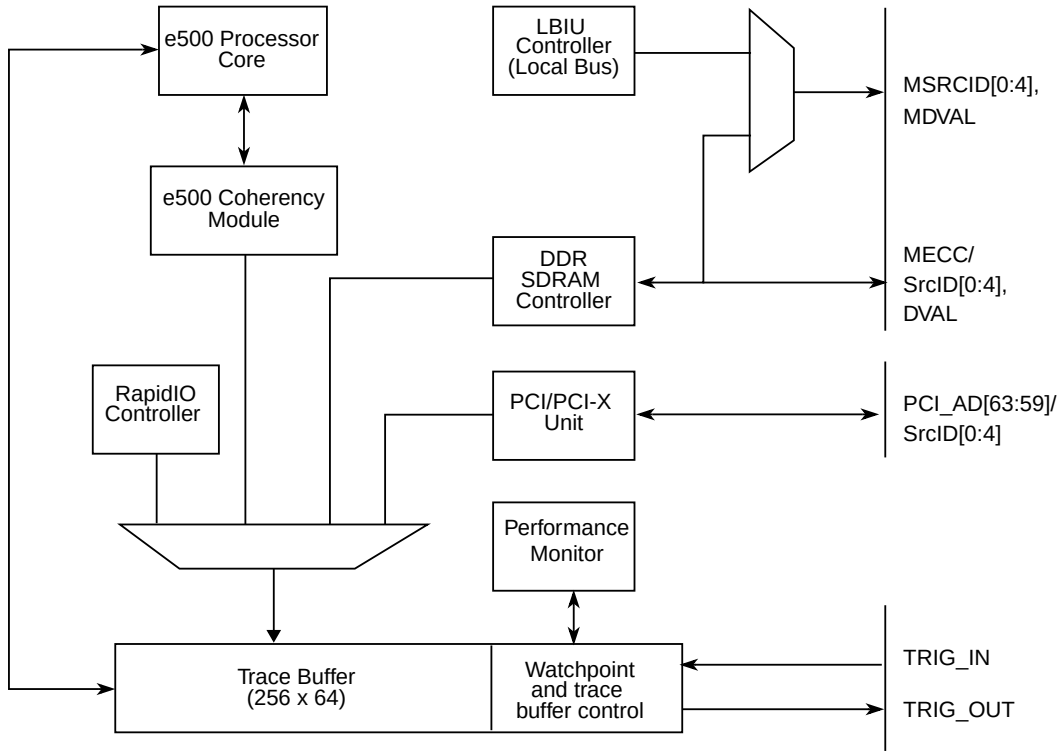


Figure 19-1. Debug and Watchpoint Monitor Block Diagram

19.1.1 Overview

As shown in Figure 19-1, 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.

19.1.2 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

19.1.3 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.

19.1.3.1 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.

19.1.3.2 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.

19.1.3.3 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 $\overline{\text{PCI_GNT3}}$. If $\overline{\text{PCI_GNT3}}$ equals 0, then the source ID information is driven onto ADDR[63:59].

19.1.3.4 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.

19.1.3.5 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.

19.2 External Signal Description

19.2.1 Overview

All the signals associated with the features described in this document are listed in Table 19-1. 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 19-1. 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 19-24 for a description of these functions. This is an output.	1	–

Table 19-1. 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 asynchrounously.		
TEST_SEL	Test	Factory Test		
LSSD_MODE	Test	Factory Test		
L1_TSTCLK	Test	Factory Test		
L2_TSTCLK	Test	Factory Test		

19.2.2 Detailed Signal Descriptions

19.2.2.1 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.

19.2.2.2 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.

19.2.2.3 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 MPC8560. 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.

19.2.2.4 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.

19.2.2.4.1 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 MPC8560 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 MPC8560 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.

19.2.2.4.2 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.

19.2.2.5 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 MPC8560.

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.

19.2.3 Test and Configuration Signals

The MPC8560 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 MPC8560 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.

19.2.3.1 JTAG Test Clock (TCK)—Input

The JTAG test clock (TCK) signal is an input on the MPC8560. 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.

19.2.3.2 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.

19.2.3.3 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.

19.2.3.4 JTAG Test Mode Select (TMS)—Input

The test mode select (TMS) signal is an input on the MPC8560. 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.

19.2.3.5 JTAG Test Reset ($\overline{\text{TRST}}$)—Input

The test reset ($\overline{\text{TRST}}$) signal is an input on the MPC8560. 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 MPC8560.

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.

19.2.3.6 $\overline{\text{LSSD_MODE}}$—Input

$\overline{\text{LSSD_MODE}}$ is used for factory test.

19.2.3.7 L1_TSTCLK—Input

L1_TSTCLK is used for factory test.

19.2.3.8 L2_TSTCLK—Input

L2_TSTCLK is used for factory test.

19.2.3.9 $\overline{\text{TEST_SEL}}$—Input

$\overline{\text{TEST_SEL}}$ is used for factory test and should be tied to binary 1.

19.3 Memory Map/Register Definition

Table 19-2 shows the memory mapped registers accessible through the internal register interface.

Table 19-2. 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	19.3.1.1/19-11
0xE_2004	Watchpoint Monitor Control Register 1 (WMCR1)	R/W	0x0000_0000	19.3.1.2/19-12
0xE_200C	Watchpoint Monitor Address Register (WMAR)	R/W	0x0000_0000	19.3.1.3/19-13
0xE_2014	Watchpoint Monitor Address Mask Register (WMAMR)	R/W	0x0000_0000	19.3.1.4/19-14
0xE_2018	Watchpoint Monitor Transaction Mask Register (WMTMR)	R/W	0x0000_0000	19.3.1.5/19-15
0xE_201C	Watchpoint Monitor Status Register (WMSR)	R/W	0x0000_0000	19.3.1.6/19-17
Trace Buffer Registers				
0xE_2040	Trace Buffer Control Register (TBCR0)	R/W	0x0000_0000	19.3.1.1/19-11
0xE_2044	Trace Buffer Control Register (TBCR1)	R/W	0x0000_0000	19.3.1.2/19-12
0xE_204C	Trace Buffer Address Register (TBAR)	R/W	0x0000_0000	19.3.2.3/19-20
0xE_2054	Trace Buffer Address Mask Register (TBAMR)	R/W	0x0000_0000	19.3.2.4/19-21
0xE_2058	Trace Buffer Transaction Mask Register (TBTMR)	R/W	0x0000_0000	19.3.2.5/19-22
0xE_205C	Trace Buffer Status Register (TBSR)	R/W	0x0000_0000	19.3.2.6/19-22

Table 19-2. Debug and Watchpoint Monitor Memory Map (continued)

Local Memory Offset	Register	Access	Reset	Section/Page
0xE_2060	Trace Buffer Access Control Register (TBACR)	R/W	0x0000_0000	19.3.2.7/19-23
0xE_2064	Trace Buffer Access Data High Register (TBADHR)	R/W	0x0000_0000	19.3.2.8/19-24
0xE_2068	Trace Buffer Access Data Register (TBADR)	R/W	0x0000_0000	19.3.2.9/19-25
Context ID Registers				
0xE_20A0	Programmed Context ID Register (PCIDR)	R/W	0x0000_0000	19.3.3.1/19-25
0xE_20A4	Current Context ID Register (CCIDR)	R/W	0x0000_0000	19.3.3.2/19-26
Other Registers				
0xE_20B0	Trigger Output Source Register (TOSR)	R/W	0x0000_0000	19.3.4.1/19-27

19.3.1 Watchpoint Monitor Register Descriptions

19.3.1.1 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.

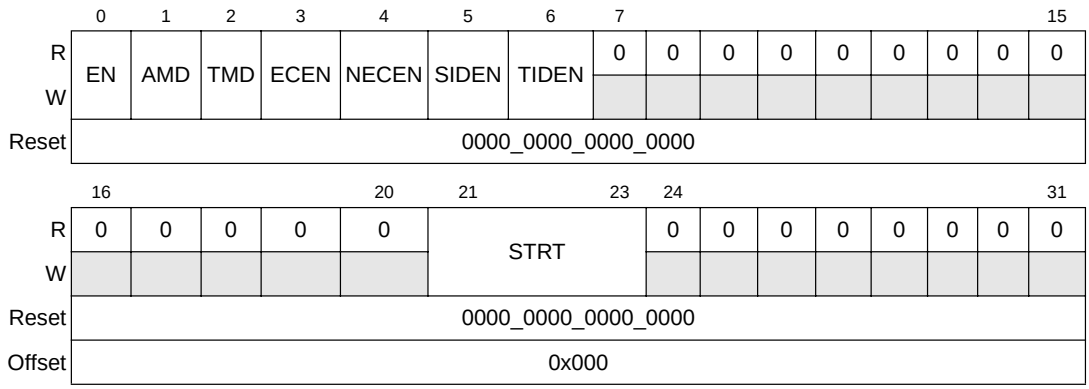


Figure 19-2. Watchpoint Monitor Control Register 0 (WMCR0)

Table 19-3. 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 19-3. 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 programed context ID 111 Reserved
24–31	—	Reserved

19.3.1.2 Watchpoint Monitor Control Register 1 (WMCR1)

This register contains the other part of the controls for setting up a watchpoint monitor event.

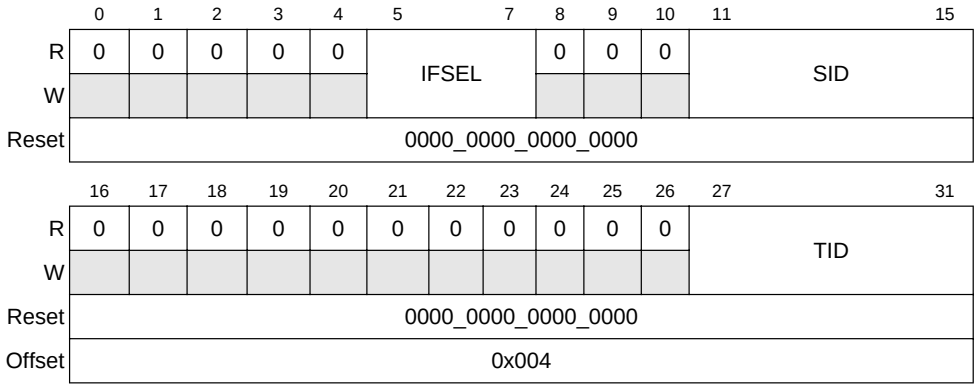


Figure 19-3. Watchpoint Monitor Control Register (WMCR1)

Table 19-4. 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 19-25.
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 19-25.

19.3.1.3 Watchpoint Monitor Address Register (WMAR)

This register contains the address to match against if the WMCR[AMD] = 0.

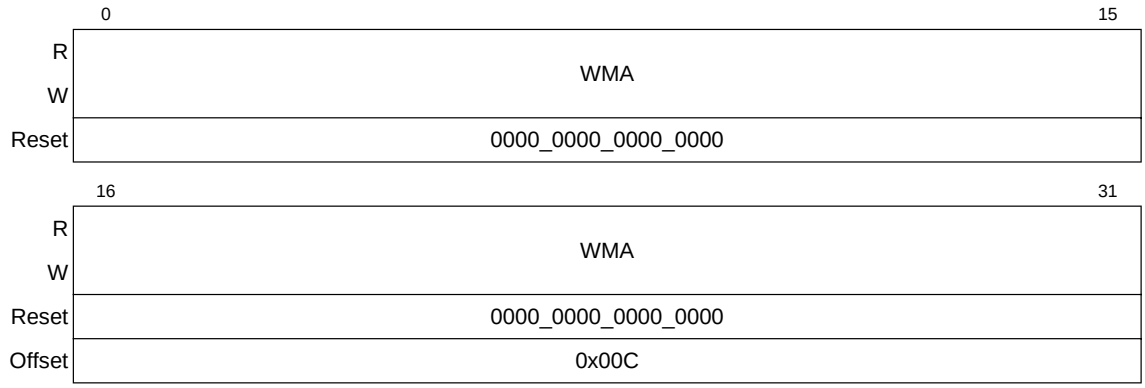


Figure 19-4. Watchpoint Monitor Address Register (WMAR)

Table 19-5. 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.

19.3.1.4 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.

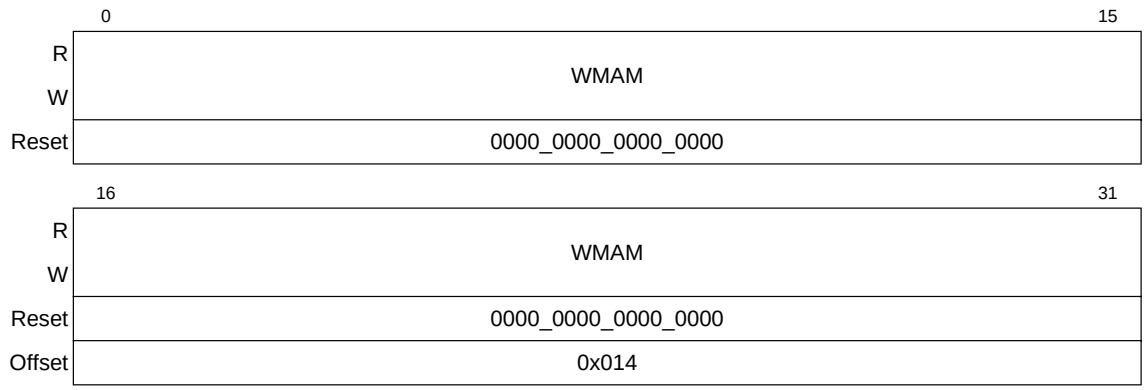


Figure 19-5. Watchpoint Monitor Address Mask Register (WMAMR)

Table 19-6. WMAMR Field Descriptions

Bits	Name	Description
0–31	WMAM	Watchpoint monitor address mask. Low order bits of the match address mask.

19.3.1.5 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.

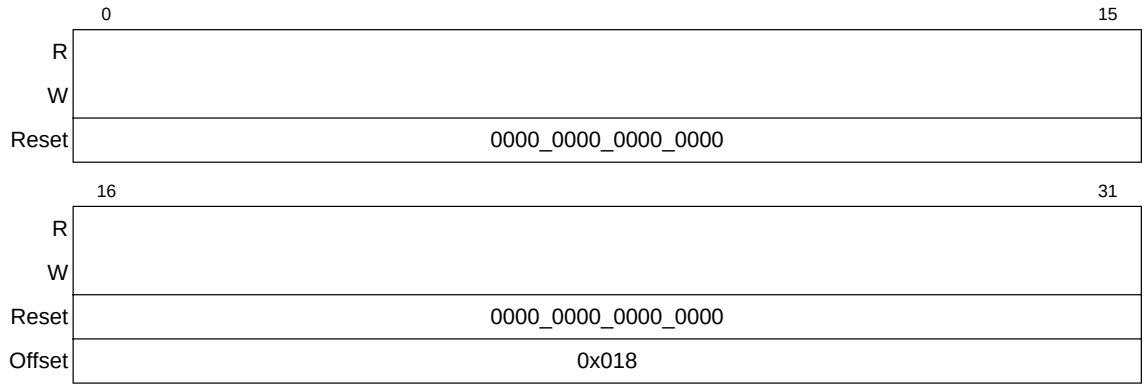


Figure 19-6. Watchpoint Monitor Transaction Mask Register (WMTMR)

Table 19-7. 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 19-8. 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 19-8. 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 19-9. 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 19-10. 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 19-11. 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

19.3.1.6 Watchpoint Monitor Status Register (WMSR)

This register contains the status of the watchpoint monitor.

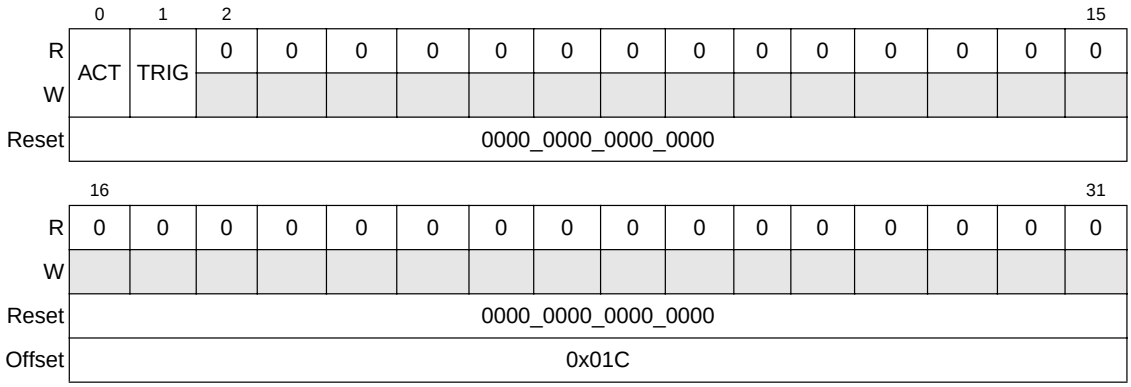


Figure 19-7. Watchpoint Monitor Status Register (WMSR)

Table 19-12. 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

19.3.2 Trace Buffer Register Descriptions

19.3.2.1 Trace Buffer Control Register (TBCR0)

This register contains part of the controls for setting up a trace buffer event. See Section 19.3.2.2, “Trace Buffer Control Register 1 (TBCR1),” for other trace buffer event controls.

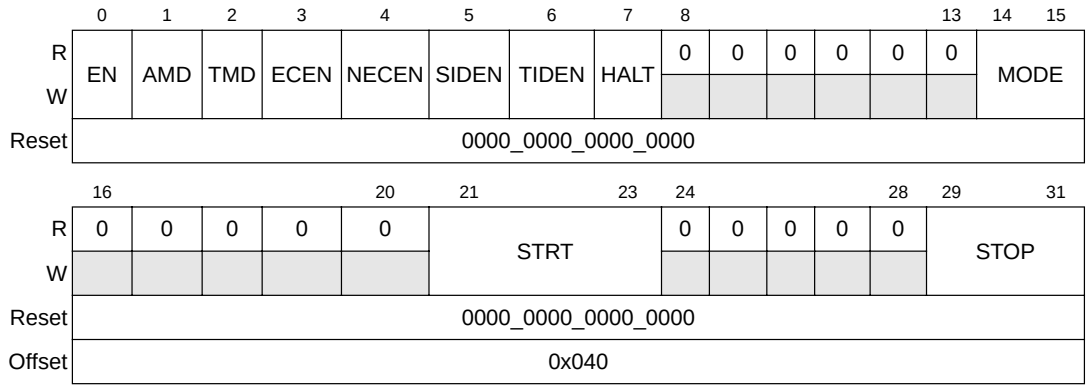


Figure 19-8. Trace Buffer Control Register 0 (TBCR0)

Table 19-13. 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 19-13. 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 programed 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 programed 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.

19.3.2.2 Trace Buffer Control Register 1 (TBCR1)

This register contains a portion of the controls for setting up a trace buffer event. See Section 19.3.2.1, “Trace Buffer Control Register (TBCR0)” for other trace buffer event controls.



Figure 19-9. Trace Buffer Control Register (TBCR1)

Table 19-14. 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 19-25.
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 19-25.

19.3.2.3 Trace Buffer Address Register (TBAR)

This register contains the address to match against if TBCR[AMD] = 0.

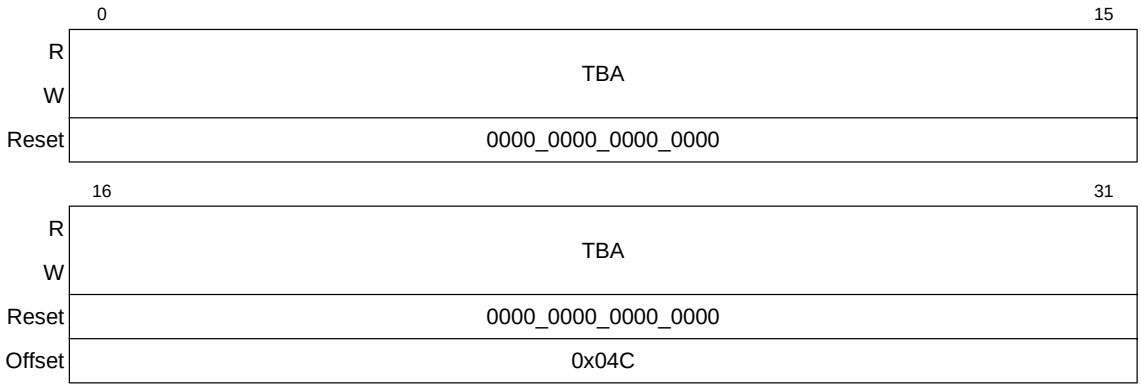


Figure 19-10. Trace Buffer Address Register (TBAR)

Table 19-15. TBAR Field Descriptions

Bits	Name	Description
0–31	TBA	Trace buffer address. Low order bits of the match address

19.3.2.4 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.

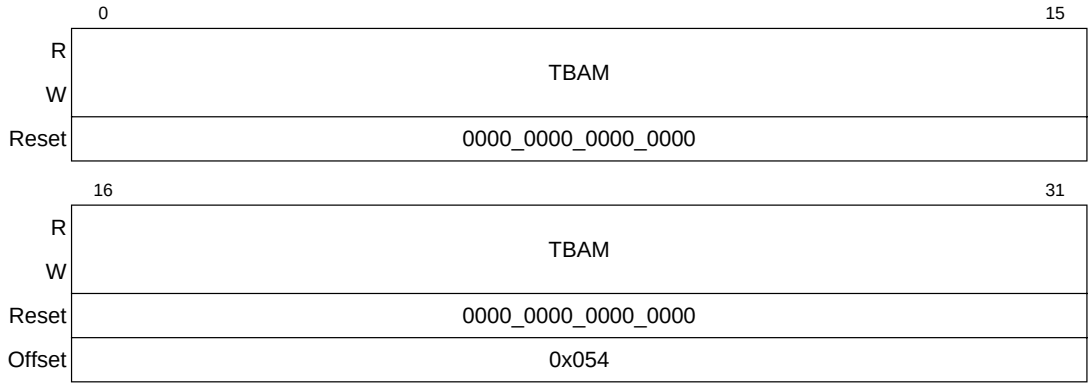


Figure 19-11. Trace Buffer Address Mask Register (TBAMR)

Table 19-16. TBAMR Field Descriptions

Bits	Name	Description
0–31	TBAM	Trace buffer address mask. Low order bits of the match address mask.

19.3.2.5 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.

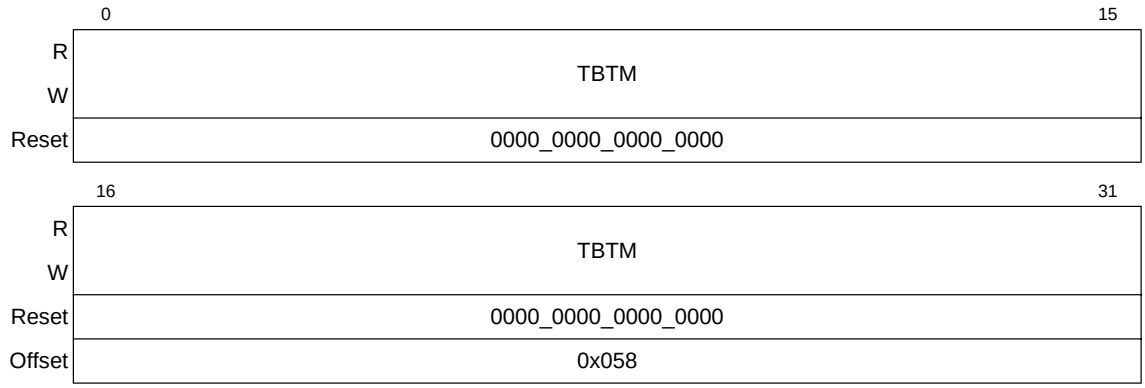


Figure 19-12. Trace Buffer Transaction Mask Register (TBTMR)

Table 19-17. 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 19-8 through Table 19-11 for the definition of the transaction types.

19.3.2.6 Trace Buffer Status Register (TBSR)

This register contains the status of the trace buffer.

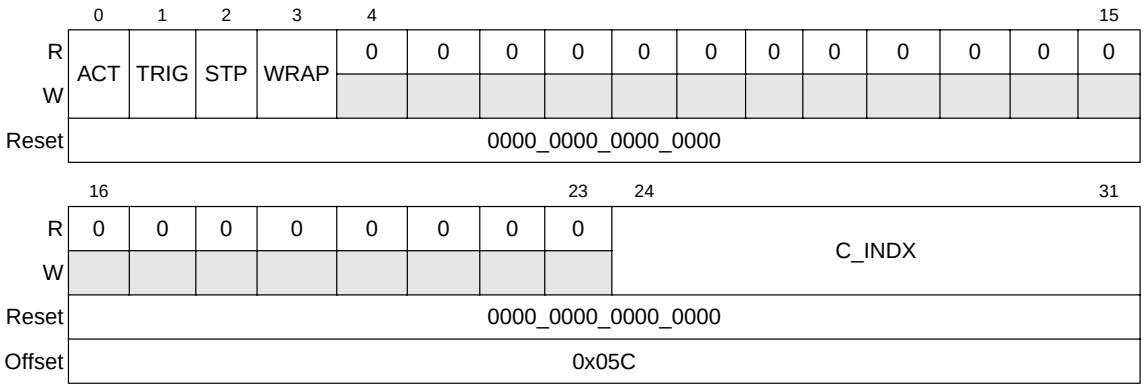


Figure 19-13. Trace Buffer Status Register (TBSR)

Table 19-18. 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_INDX	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.

19.3.2.7 Trace Buffer Access Control Register (TBACR)

The trace buffer access control register (TBACR) enables sofware 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.

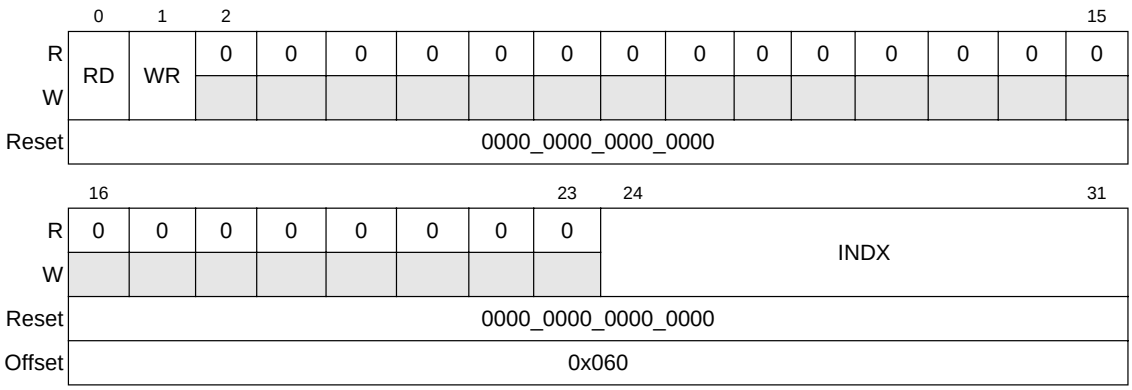


Figure 19-14. Trace Buffer Access Control Register (TBACR)

Table 19-19. 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 occures 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].

19.3.2.8 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.

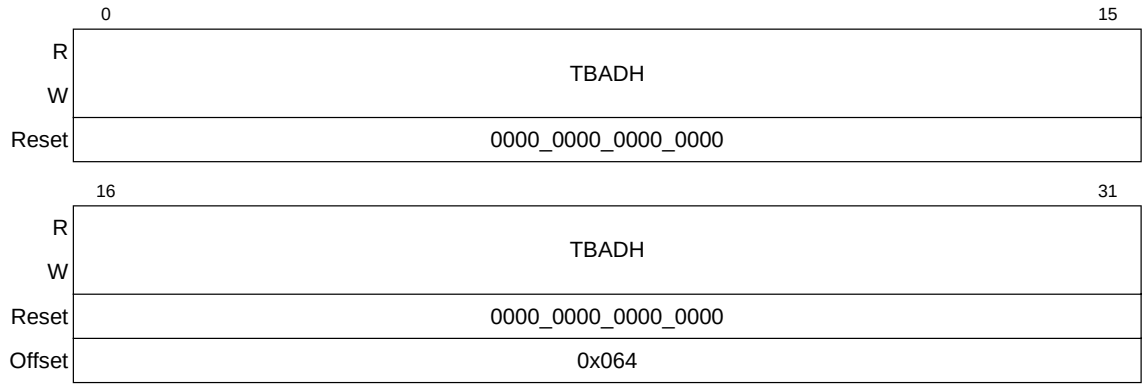


Figure 19-15. Trace Buffer Read High Register (TBADHR)

Table 19-20. 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.

19.3.2.9 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.

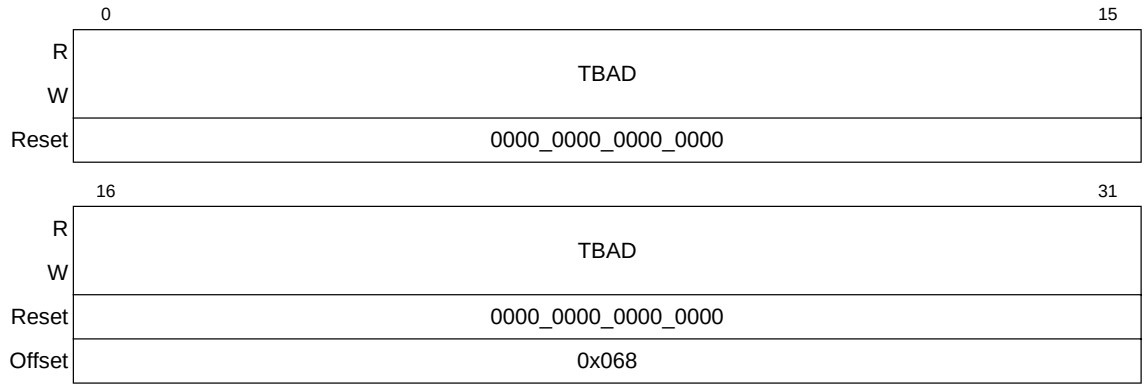


Figure 19-16. Trace Buffer Read High Register (TBADR)

Table 19-21. 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.

19.3.3 Context ID Registers

19.3.3.1 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).

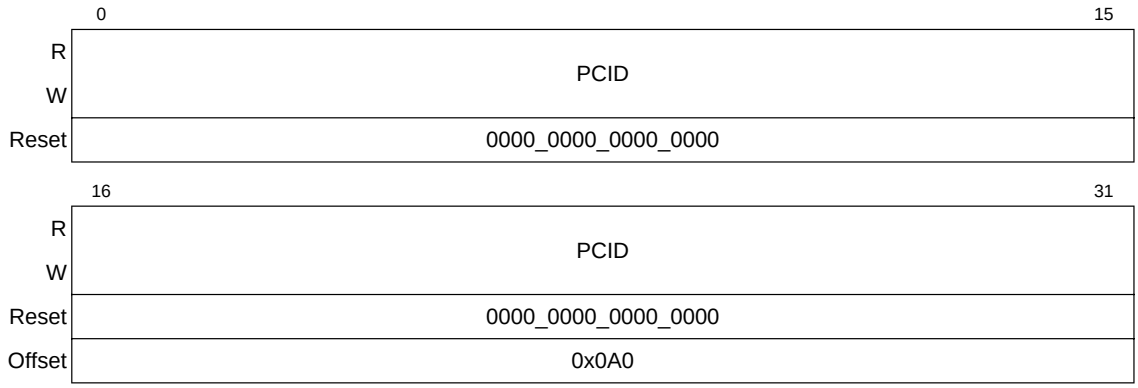


Figure 19-17. Programmed Context ID Register (PCIDR)

Table 19-22. PCIDR Field Descriptions

Bits	Name	Description
0–31	PCID	Programmed Context ID

19.3.3.2 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).

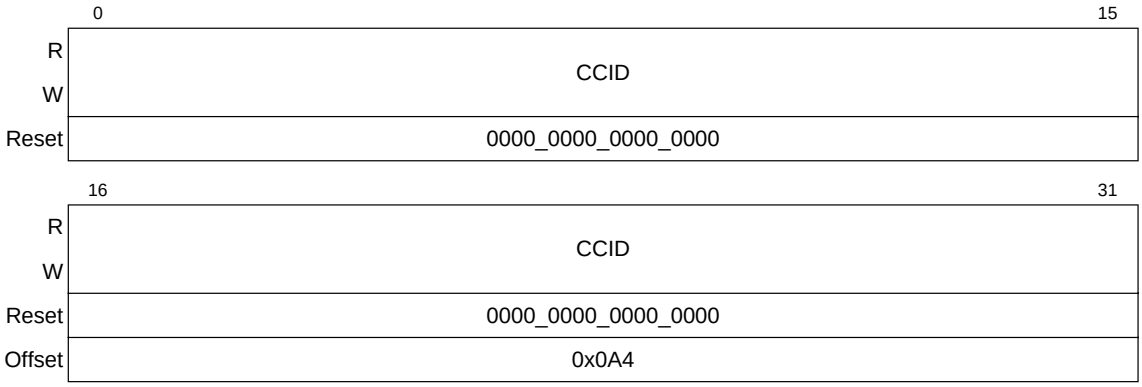


Figure 19-18. Current Context ID Register (CCIDR)

Table 19-23. CCIDR Field Descriptions

Bits	Name	Description
0–31	CCID	Current Context ID

19.3.4 Trigger Out Function

19.3.4.1 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.

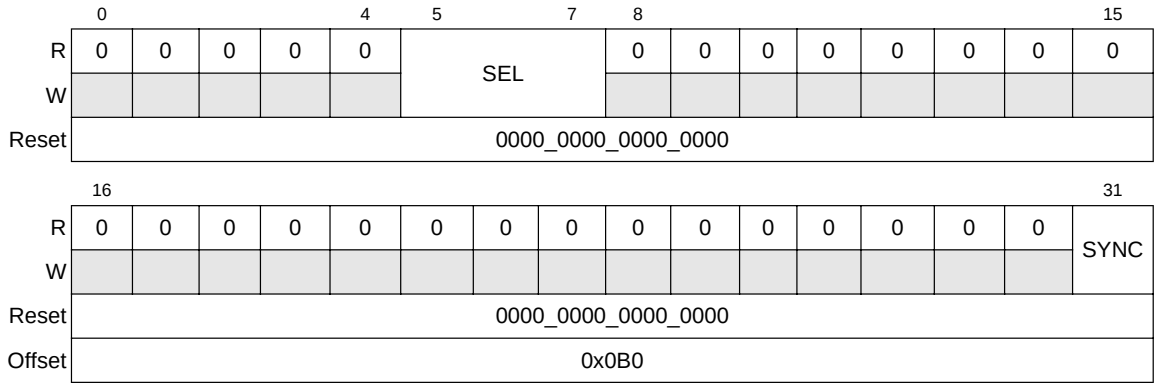


Figure 19-19. Trigger Out Source Register (TOSR)

Table 19-24. 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.

19.4 Functional Description

The debug features on the MPC8560 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 19-25 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,

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 19-25. 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	CPM
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 19-25. 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)

19.4.1 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 (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 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 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.

19.4.2 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.

19.4.2.1 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.

19.4.2.2 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.

19.4.3 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.

19.4.4 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.

19.4.4.1 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 18, "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.

19.4.5 Trace Buffer

The trace buffer is a 256×64 array that can be used fto 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 (TBCR*n*) and the watchpoint monitor control registers (WMCR*n*) shows that the trace buffer controls that are not needed for the watchpoint monitor are marked reserved in WMCR*n*. This permits using the trace buffer as a second watchpoint monitor by simply ignoring the trace options.

The trace buffer 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.

19.4.5.1 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 19-26. 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 19-8 on page 19-15. 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 19-25 on page 19-28. 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 19-25 on page 19-28 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 19-27. 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 19-9 on page 19-16. 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 19-25 on page 19-28. 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 19-28. 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 19-10 on page 19-16. 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 19-25 on page 19-28. 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 19-28. Buffer Entry Definition for TBCR[TRSEL] = 0b010 (PCI)

Bits	Name	Function
12–31		Reserved
32–63		Address bits 0–31

Table 19-29. 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 19-11 on page 19-17. 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 19-25 on page 19-28. 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

19.5 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.

Part V
CPM Features

The MPC8560 communications processor module (CPM) is a superset of the MPC8260 PowerQUICC II CPM, with enhancements in performance and the addition of hardware and microcode routines for supporting high bit-rate protocols like ATM and fast Ethernet. The support for multiple high-level data link control (HDLC) channels is enhanced to support up to 256 HDLC channels. This part defines the CPM blocks of the MPC8560. It contains the following chapters:

- Chapter 20, “Communications Processor Module Overview,” provides a high-level summary of the MPC8560’s features and memory map.
- Chapter 21, “CPM Interrupt Controller,” describes the CPM interrupt controller of the MPC8560.
- Chapter 22, “Serial Interface with Time-Slot Assigner,” describes the serial interface and TSA of the MPC8560.
- Chapter 23, “CPM Multiplexing,” describes how the CPM multiplexing logic (CMX) connects the physical layer (UTOPIA, MII, modem lines, TDM lines and proprietary serial lines) to the FCCs and SCCs.
- Chapter 24, “Baud-Rate Generators (BRGs),” describes the eight independent, identical baud-rate generators (BRGs) that can be used with the FCCs and SCCs.
- Chapter 25, “CPM Timers,” describes the four identical 16-bit general-purpose CPM timers that can alternately be used as two 32-bit timers.
- Chapter 26, “SDMA Channels,” describes the two physical serial DMA (SDMA) channels of the MPC8560.
- Chapter 27, “Serial Communications Controllers (SCCs),” describes the four serial communications controllers (SCCs) which can be configured independently to implement different protocols for bridging functions, routers, and gateways, and to interface with a wide variety of standard WANs, LANs, and proprietary networks..
- Chapter 28, “SCC UART Mode,” describes how the general SCC mode register (GSMR) is used to configure an SCC channel to function in UART mode, which

provides standard serial I/O using asynchronous character-based (start-stop) protocols with RS-232C-type lines.

- Chapter 29, “SCC HDLC Mode,” describes how HDLC mode is selected for an SCC. In HDLC mode, an SCC becomes an HDLC controller, and consists of separate transmit and receive sections whose operations are asynchronous with the core and can either be synchronous or asynchronous with respect to other SCCs.
- Chapter 30, “SCC BISYNC Mode,” describes how transparent BISYNC mode allows full binary data to be sent with any possible character pattern.
- Chapter 31, “SCC Transparent Mode,” describes how an SCC in transparent mode functions as a high-speed serial-to-parallel and parallel-to-serial converter.
- Chapter 32, “SCC AppleTalk Mode,” describes how the MPC8560 provides LocalTalk protocol support. The AppleTalk controller provides required frame synchronization, bit sequence, preamble, and postamble onto standard HDLC frames.
- Chapter 33, “Multi-Channel Controllers (MCCs),” describes the MPC8560’s two multi-channel controllers (MCC1 and MCC2) and how each handles up to 128 serial, full-duplex data channels.
- Chapter 34, “Fast Communications Controllers (FCCs),” describes how the FCCs can be configured independently to implement different protocols. Together, they can be used to implement bridging functions, routers, and gateways, and to interface with a wide variety of standard WANs, LANs, and proprietary networks.
- Chapter 35, “ATM Controller,” describes the ATM controller that provides the ATM and AAL layers of the ATM protocol using the universal test and operations physical layer (PHY) interface for ATM (UTOPIA level II) for both master and slave modes.
- Chapter 36, “AAL1 Circuit Emulation Service,” describes the implementation of circuit emulation service (CES microcode package) using AAL1 on the MPC8560.
- Chapter 37, “AAL2,” describes the AAL2 microcode package.
- Chapter 38, “Transmission Convergence Layer,” describes how the MPC8560 can support applications that receive ATM traffic over the standard serial protocols like E1, T1, and xDSL via its serial interface (SIx TDMx and NMSI) ports because the ATM transmission convergence (TC) layer functionality is implemented internally.
- Chapter 39, “CPM Ethernet Controller,” describes the fast Ethernet controller in the CPM.
- Chapter 40, “FCC HDLC Controller,”describes the FCC HDLC controller of the MPC8560.
- Chapter 41, “FCC Transparent Controller,” describes how the FCC transparent controller functions as a high-speed serial-to-parallel and parallel-to-serial converter.

- Chapter 42, “Serial Peripheral Interface (SPI),” describes the serial peripheral interface (SPI) of the MPC8560 CPM.
- Chapter 43, “CPM I^2C Controller,” describes the I^2C controller of the CPM.
- Chapter 44, “Parallel I/O Ports,” describes the four general-purpose I/O ports of the CPM.
- Chapter 45, “CPM ROM Microcode: SS7 Controller,” describes the SS7 RAM microcode package.

Chapter 20
Communications Processor Module Overview

The MPC8560 communications processor module (CPM) is a superset of the MPC8260 PowerQUICC II CPM, with enhancements in performance and the addition of hardware and microcode routines for supporting high bit-rate protocols like ATM and Fast Ethernet. The support for multiple high-level data link control (HDLC) channels is enhanced to support up to 256 HDLC channels.

20.1 Features

The CPM includes various blocks to provide the system with an efficient way to handle data communication tasks. The following is a list of the CPM's important features.

- Communications processor (CP)
  - — One instruction per clock
  - — Executes code from internal ROM or instruction RAM
  - — 32-bit RISC architecture
  - — Tuned for communication environments: instruction set supports CRC computation and bit manipulation.
  - — Internal timer
  - — Interfaces with the embedded e500 core processor through a 32-Kbyte dual-port RAM and virtual DMA channels for each peripheral controller
  - — Handles serial protocols and virtual DMA.
- Three full-duplex fast serial communications controllers (FCCs) support the following protocols:
  - — ATM protocol through UTOPIA interface (FCC1 and FCC2 only)
  - — IEEE802.3/Fast Ethernet
  - — HDLC
  - — Totally transparent operation

- Two multi-channel controllers (MCCs) that together can handle up to 256 HDLC/transparent channels at 64 Kbps each, multiplexed on up to eight TDM interfaces
- Four full-duplex serial communications controllers (SCCs) support the following protocols:
  - High level/synchronous data link control (HDLC/SDLC)
  - LocalTalk (HDLC-based local area network protocol)
  - Universal asynchronous receiver transmitter (UART)
  - Synchronous UART (1x clock mode)
  - Binary synchronous communication (BISYNC)
  - Totally transparent operation
- Serial peripheral interface (SPI) support for master or slave
- I^2C bus controller
- Time-slot assigner supports multiplexing of data from any of the SCCs and FCCs onto eight time-division multiplexed (TDM) interfaces. The time-slot assigner supports the following TDM formats:
  - T1/CEPT lines
  - T3/E3
  - Pulse code modulation (PCM) highway interface
  - ISDN primary rate
  - Motorola interchip digital link (IDL)
  - General circuit interface (GCI)
  - User-defined interfaces
- Eight independent baud rate generators (BRGs)
- Four general-purpose 16-bit timers or two 32-bit timers
- General-purpose parallel ports—16 parallel I/O lines with interrupt capability

Figure 20-1 shows the MPC8560 CPM block diagram.

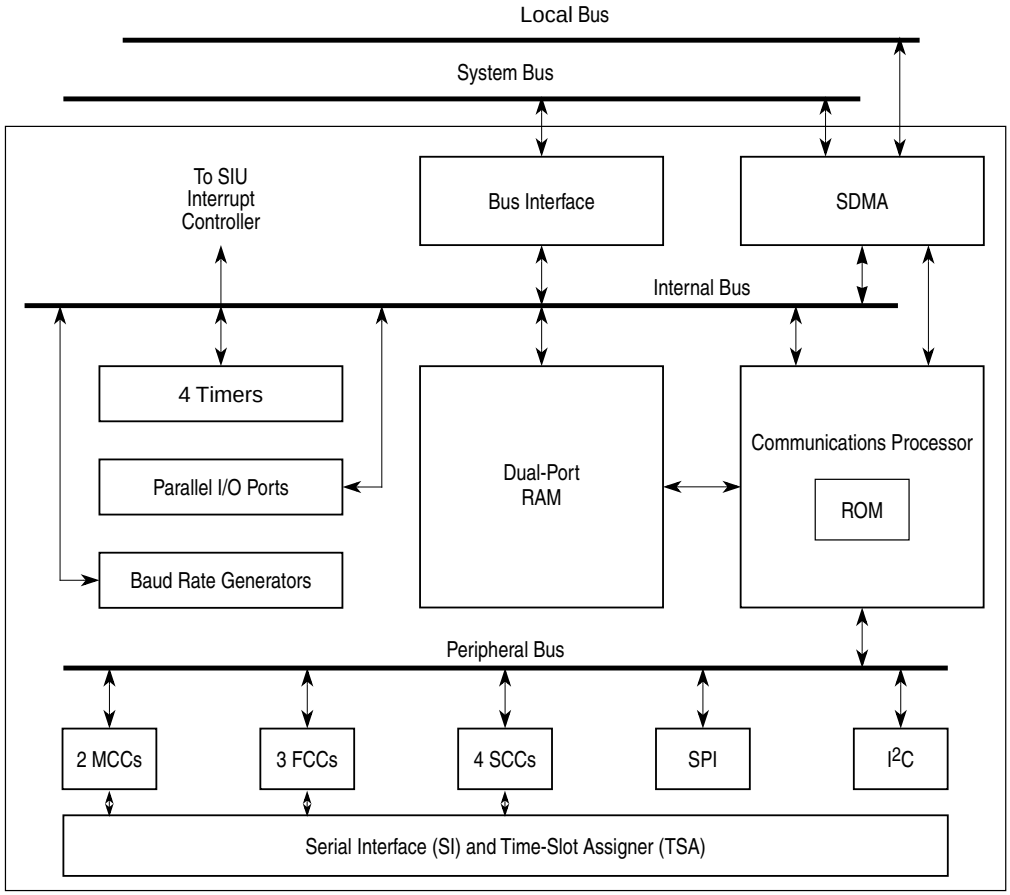


Figure 20-1. MPC8560 CPM Block Diagram

20.1.1 CPM Memory Map

Table 20-1 shows the CPM portion of the internal memory map.

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
CPM Dual-Port RAM				
0x8_0000–0x8_3FFF	Dual-port RAM (DPRAM1)			20.5/-31
0x8_4000–0x8_7FFF	Reserved	—	—	—
0x8_8000–0x8_BFFF	Dual-port RAM (DPRAM2)			20.5/-31
0x8_C000–0x8_FFFF	Reserved	—	—	—

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
General SIU				
0x9_0050	System bus address error register (SMAER)	R		26.1.3/26-3
0x9_0054	System bus status error register (SMSER)	R		26.1.4/26-3
0x9_0058	System bus event register (SMEVR)	R/W	—	27.1.5/27-4
0x9_005C–0x9_005F	Reserved	—		—
0x9_0060	Local bus address error register (LMAER)	R		27.1.3/27-3
0X9_0064	Local bus status error register (LMSER)	R		27.1.4/27-3
0x9_0068	Local bus event register (LMEVR)	R/W		27.1.5/27-4
0x9_006c–0x9_00FF	Reserved	—	—	—
Interrupt Controller				
0x9_0C00	SIU interrupt configuration register (SICR)	R/W	0x0000_0000	0.1.1.1/4-20
0x9_0C02	Reserved	—	—	—
0x9_0C04	SIU interrupt vector register (SIVEC)	R/W	0x0000_0000	0.1.1.6/4-27
0x9_0C08	SIU interrupt pending register (high) (SIPNR_H)	R/W	0x0000_0000	0.1.1.4/4-24
0x9_0C0C	SIU interrupt pending register (low) (SIPNR_L)	R/W	0x0000_0000	0.1.1.4/4-24
0x9_0C10	Reserved	—	—	—
0x9_0C14	CPM interrupt priority register (high) (SCPRR_H)	R/W	0x0530_9770	0.1.1.3/4-22
0x9_0C18	CPM interrupt priority register (low) (SCPRR_L)	R/W	0x0530_9770	0.1.1.3/4-22
0x9_0C1C	SIU interrupt mask register (high) (SIMR_H)	R/W	0x0000_0000	0.1.1.5/4-26
0x9_0C20	SIU interrupt mask register (low) (SIMR_L)	R/W	0x0000_0000	0.1.1.5/4-26
0x9_0C24	SIU external interrupt control register (SIEXR)	R/W	0x0000_0000	0.1.1.7/4-28
0x9_0C28–0x9_0C7F	Reserved	—	—	—
0x9_0C80	System Clock Control Register (SCCR)	R/W	0x0000_0000	
Input/Output Port				
0x9_0D00	Port A data direction register (PDIRA)	R/W	0x0000_0000	42.2.3/-3
0x9_0D04	Port A pin assignment register (PPARA)	R/W	0x0000_0000	42.2.4/-4
0x9_0D08	Port A special options register (PSORA)	R/W	0x0000_0000	42.2.5/-4
0x9_0D0C	Port A open drain register (PODRA)	R/W	0x0000_0000	42.2.1/-2
0x9_0D10	Port A data register (PDATA)	R/W	0x0000_0000	42.2.2/-2
0x9_0D14–0x9_0D1F	Reserved	—	—	—

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_0D20	Port B data direction register (PDIRB)	R/W	0x0000_0000	42.2.3/-3
0x9_0D24	Port B pin assignment register (PPARB)	R/W	0x0000_0000	42.2.4/-4
0x9_0D28	Port B special operation register (PSORB)	R/W	0x0000_0000	42.2.5/-4
0x9_0D2C	Port B open drain register (PODRB)	R/W	0x0000_0000	42.2.1/-2
0x9_0D30	Port B data register (PDATB)	R/W	0x0000_0000	42.2.2/-2
0x9_0D34–0x9_0D3F	Reserved	—	—	—
0x9_0D40	Port C data direction register (PDIRC)	R/W	0x0000_0000	42.2.3/-3
0x9_0D44	Port C pin assignment register (PPARC)	R/W	0x0000_0000	42.2.4/-4
0x9_0D48	Port C special operation register (PSORC)	R/W	0x0000_0000	42.2.5/-4
0x9_0D4C	Port C open drain register (PODRC)	R/W	0x0000_0000	42.2.1/-2
0x9_0D50	Port C data register (PDATC)	R/W	0x0000_0000	42.2.2/-2
0x9_0D54–0x9_0D5F	Reserved	—	—	—
0x9_0D60	Port D data direction register (PDIRD)	R/W	0x0000_0000	42.2.3/-3
0x9_0D64	Port D pin assignment register (PPARD)	R/W	0x0000_0000	42.2.4/-4
0x9_0D68	Port D special operation register (PSORD)	R/W	0x0000_0000	42.2.5/-4
0x9_0D6C	Port D open drain register (PODRD)	R/W	0x0000_0000	42.2.1/-2
0x9_0D70	Port D data register (PDATD)	R/W	0x0000_0000	42.2.2/-2
CPM Timers				
0x9_0D80	Timer 1 and timer 2 global configuration register (TGCR1)	R/W	0x00	25.2.2/-4
0x9_0D81	Reserved	—	—	—
0x9_0D84	Timer 3 and timer 4 global configuration register (TGCR2)	R/W	0x00	25.2.2/-4
0x9_0D85–0x9_0D8F	Reserved	—	—	—
0x9_0D90	Timer 1 mode register (TMR1)	R/W	0x0000	25.2.3/-6
0x9_0D92	Timer 2 mode register (TMR2)	R/W	0x0000	25.2.3/-6
0x9_0D94	Timer 1 reference register (TRR1)	R/W	0x0000	25.2.4/-7
0x9_0D96	Timer 2 reference register (TRR2)	R/W	0x0000	25.2.4/-7
0x9_0D98	Timer 1 capture register (TCR1)	R/W	0x0000	25.2.5/-8
0x9_0D9A	Timer 2 capture register (TCR2)	R/W	0x0000	25.2.5/-8
0x9_0D9C	Timer 1 counter (TCN1)	R/W	0x0000	25.2.6/-8
0x9_0D9E	Timer 2 counter (TCN2)	R/W	0x0000	25.2.6/-8
0x9_0DA0	Timer 3 mode register (TMR3)	R/W	0x0000	25.2.3/-6

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_0DA2	Timer 4 mode register (TMR4)	R/W	0x0000	25.2.3/-6
0x9_0DA4	Timer 3 reference register (TRR3)	R/W	0x0000	25.2.4/-7
0x9_0DA6	Timer 4 reference register (TRR4)	R/W	0x0000	25.2.4/-7
0x9_0DA8	Timer 3 capture register (TCR3)	R/W	0x0000	25.2.5/-8
0x9_0DAA	Timer 4 capture register (TCR4)	R/W	0x0000	25.2.5/-8
0x9_0DAC	Timer 3 counter (TCN3)	R/W	0x0000	25.2.6/-8
0x9_0DAE	Timer 4 counter (TCN4)	R/W	0x0000	25.2.6/-8
0x9_0DB0	Timer 1 event register (TER1)	R/W	0x0000	25.2.7/-8
0x9_0DB2	Timer 2 event register (TER2)	R/W	0x0000	25.2.7/-8
0x9_0DB4	Timer 3 event register (TER3)	R/W	0x0000	25.2.7/-8
0x9_0DB6	Timer 4 event register (TER4)	R/W	0x0000	25.2.7/-8
0x9_0DB8–0x9_1017	Reserved	—	—	—
SDMA				
0x9_1018	SDMA status register (SDSR)		0x00	26.1.1/-2
0x9_1019	Reserved	—	—	—
0x9_101C	SDMA mask register (SDMR)	R/W	0x00	26.1.2/-3
0x9_101D	Reserved	—	—	—
FCC1				
0x9_1300	FCC1 general mode register (GFMR1)	R/W	0x0000_0000	34.2/-3
0x9_1304	FCC1 protocol-specific mode register (FPSMR1)	R/W	0x0000_0000	(ATM) 35.13.2/-87 (Ethernet) 37.19.1/-21 (HDLC) 38.6/-8
0x9_1308	FCC1 transmit on demand register (FTODR1)	R/W	0x0000	34.5/-8
0x9_130A	Reserved	—	—	—
0x9_130C	FCC1 data synchronization register (FDSR1)	R/W	0x7E7E	34.4/-7
0x9_130E	Reserved	—	—	—
0x9_1310	FCC1 event register (FCCE1)	R/W	0x0000_0000	(ATM) 35.13.3/-89 (Ethernet) 37.19.2/-23 (HDLC) 38.9/-14
0x9_1312	Reserved	—	—	—

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_1314	FCC1 mask register (FCCM1)	R/W	0x0000_0000	(ATM) 35.13.3/-89 (Ethernet) 37.19.2/-23 (HDLC) 38.9/-14
0x9_1316	Reserved	—	—	—
0x9_1318	FCC1 status register (FCCS1)	R	0x00	38.10/-16 (HDLC)
0x9_1319	Reserved	—	—	—
0x9_131C	FCC1 transmit internal rate registers for PHY0 (FTIRR1_PHY0)	R/W	0x00	35.13.4/-91 (ATM)
0x9_131D	FCC1 transmit internal rate registers for PHY1 (FTIRR1_PHY1)	R/W	0x00	
0x9_131E	FCC1 transmit internal rate registers for PHY2 (FTIRR1_PHY2)	R/W	0x00	
0x9_131F	FCC1 transmit internal rate registers for PHY3 (FTIRR1_PHY3)	R/W	0x00	
FCC2				
0x9_1320	FCC2 general mode register (GFMR2)	R/W	0x0000_0000	34.2/-3
0x9_1324	FCC2 protocol-specific mode register (FPSMR2)	R/W	0x0000_0000	(ATM) 35.13.2/-87 (Ethernet) 37.19.1/-21 (HDLC) 38.6/-8
0x9_1328	FCC2 transmit on-demand register (FTODR2)	R/W	0x0000	34.5/-8
0x9_132A	Reserved	—	—	—
0x9_132C	FCC2 data synchronization register (FDSR2)	R/W	0x7E7E	34.4/-7
0x9_132E	Reserved	—	—	—
0x9_1330	FCC2 event register (FCCE2)	R/W	0x0000_0000	(ATM) 35.13.3/-89 (Ethernet) 37.19.2/-23 (HDLC) 38.9/-14
0x9_1332	Reserved	—	—	—

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_1334	FCC2 mask register (FCCM2)	R/W	0x0000_0000	(ATM) 35.13.3/-89 (Ethernet) 37.19.2/-23 (HDLC) 38.9/-14
0x9_1336	Reserved	—	—	—
0x9_1338	FCC2 status register (FCCS2)	R	0x00	38.10/-16 (HDLC)
0x9_1339	Reserved	—	0x00	—
0x9_133C	FCC2 transmit internal rate registers for PHY0 (FTIRR2_PHY0)	R/W	0x00	35.13.4/-91 (ATM)
0x9_133D	FCC2 transmit internal rate registers for PHY1 (FTIRR2_PHY1)	R/W	0x00	
0x9_133E	FCC2 transmit internal rate registers for PHY2 (FTIRR2_PHY2)	R/W	0x00	
0x9_133F	FCC2 transmit internal rate registers for PHY3 (FTIRR2_PHY3)	R/W	0x00	
FCC3				
0x9_1340	FCC3 general mode register (GFMR3)	R/W	0x0000_0000	34.2/-3
0x9_1344	FCC3 protocol-specific mode register (FPSMR3)	R/W	0x0000_0000	(ATM) 35.13.2/-87 (Ethernet) 37.19.1/-21 (HDLC) 38.6/-8
0x9_1348	FCC3 transmit on-demand register (FTODR3)	R/W	0x0000	34.5/-8
0x9_134A	Reserved	—	—	—
0x9_134C	FCC3 data synchronization register (FDSR3)	R/W	0x7E7E	34.4/-7
0x9_134E	Reserved	—	—	—
0x9_1350	FCC3 event register (FCCE3)	R/W	0x0000_0000	(ATM) 35.13.3/-89 (Ethernet) 37.19.2/-23 (HDLC) 38.9/-14
0x9_1352	Reserved	—	—	—

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_1354	FCC3 mask register (FCCM3)	R/W	0x0000_0000	(ATM) 35.13.3/-89 (Ethernet) 37.19.2/-23 (HDLC) 38.9/-14
0x9_1356	Reserved	—	—	—
0x9_1358	FCC3 status register (FCCS3)	R	0x00	38.10/-16 (HDLC)
0x9_1359–0x9_137F	Reserved	—	—	—
FCC1 (continued)				
0x9_1380	FCC1 internal rate port enable register (FIRPER1)	R/W	0x0000_0000	
0x9_1384	FCC1 internal rate event register (FIRER1)	R/W	0x0000_0000	
0x9_1388	FCC1 internal rate selection register:HI (FIRSR1_HI)	R/W	0x0000_0000	
0x9_138C	FCC1 internal rate selection register:LO (FIRSR1_LO)	R/W	0x0000_0000	
0x9_1390	General FCC1 expansion mode register (GFEMR1)	R/W	0x00	
0x9_1391–0x9_139F	Reserved	—	—	—
FCC2 (continued)				
0x9_13A0	FCC2 internal rate port enable register (FIRPER2)	R/W	0x0000_0000	
0x9_13A4	FCC2 internal rate event register (FIRER2)	R/W	0x0000_0000	
0x9_13A8	FCC2 internal rate selection register:HI (FIRSR2_HI)	R/W	0x0000_0000	
0x9_13AC	FCC2 internal rate selection register:LO (FIRSR2_LO)	R/W	0x0000_0000	
0x9_13B0	General FCC2 expansion mode register (GFEMR2)	R/W	0x00	
0x9_13B1–0x9_13CF	Reserved	—	—	—
FCC3 (continued)				
0x9_13D0	General FCC3 expansion mode register (GFEMR3)	R/W	0x00	
0x9_13D1–0x9_13FF	Reserved	—	—	—
TC Layer 1				
0x9_1400	TC1 mode register (TCMODE1)			36.11.1.1/-8
0x9_1402	TC1 cell delineation state machine register (CDSMR1)			36.11.1.2/-10
0x9_1404	TC1 event register (TCER1)			36.11.1.3/-10

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_1406	TC1 received cells counter (TC_RCC1)			36.11.3.1/-14
0x9_1408	TC1 mask register (TCMR1)			36.11.1.4/-11
0x9_140A	TC1 filtered cells counter (TC_FCC1)			36.11.3.6/-14
0x9_140C	TC1 corrected cells counter (TC_CCC1)			36.11.3.4/-14
0x9_140E	TC1 idle cells counter (TC_ICC1)			36.11.3.5/-14
0x9_1410	TC1 transmitted cells counter (TC_TCC1)			36.11.3.2/-14
0x9_1412	TC1 error cells counter (TC_ECC1)			36.11.3.3/-14
0x9_1414	Reserved	—	—	—
TC Layer 2				
0x9_1420	TC2 mode register (TCMODE2)			36.11.1.1/-8
0x9_1422	TC2 cell delineation state machine register (CDSMR2)			36.11.1.2/-10
0x9_1424	TC2 event register (TCER2)			36.11.1.3/-10
0x9_1426	TC2 received cells counter (TC_RCC2)			36.11.3.1/-14
0x9_1428	TC2 mask register (TCMR2)			36.11.1.4/-11
0x9_142A	TC2 filtered cells counter (TC_FCC2)			36.11.3.6/-14
0x9_142C	TC2 corrected cells counter (TC_CCC2)			36.11.3.4/-14
0x9_142E	TC2 idle cells counter (TC_ICC2)			36.11.3.5/-14
0x9_1430	TC2 transmitted cells counter (TC_TCC2)			36.11.3.2/-14
0x9_1432	TC2 error cells counter (TC_ECC2)			36.11.3.3/-14
0x9_1434	Reserved	—	—	—
TC Layer 3				
0x9_1440	TC3 mode register (TCMODE3)			36.11.1.1/-8
0x9_1442	TC3 cell delineation state machine register (CDSMR3)			36.11.1.2/-10
0x9_1444	TC3 event register (TCER3)			36.11.1.3/-10
0x9_1446	TC3 received cells counter (TC_RCC3)			36.11.3.1/-14
0x9_1448	TC3 mask register (TCMR3)			36.11.1.4/-11
0x9_144A	TC3 filtered cells counter (TC_FCC3)			36.11.3.6/-14
0x9_144C	TC3 corrected cells counter (TC_CCC3)			36.11.3.4/-14
0x9_144E	TC3 idle cells counter (TC_ICC3)			36.11.3.5/-14
0x9_1450	TC3 transmitted cells counter (TC_TCC3)			36.11.3.2/-14
0x9_1452	TC3 error cells counter (TC_ECC3)			36.11.3.3/-14
0x9_1454	Reserved	—	—	—

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
TC Layer 4				
0x9_1460	TC4 mode register (TCMODE4)			36.11.1.1/-8
0x9_1462	TC4 cell delineation state machine register (CDSMR4)			36.11.1.2/-10
0x9_1464	TC4 event register (TCER4)			36.11.1.3/-10
0x9_1466	TC4 received cells counter (TC_RCC4)			36.11.3.1/-14
0x9_1468	TC4 mask register (TCMR4)			36.11.1.4/-11
0x9_146A	TC4 filtered cells counter (TC_FCC4)			36.11.3.6/-14
0x9_146C	TC4 corrected cells counter (TC_CCC4)			36.11.3.4/-14
0x9_146E	TC4 idle cells counter (TC_ICC4)			36.11.3.5/-14
0x9_1470	TC4 transmitted cells counter (TC_TCC4)			36.11.3.2/-14
0x9_1472	TC4 error cells counter (TC_ECC4)			36.11.3.3/-14
0x9_1474	Reserved	—	—	—
TC Layer 5				
0x9_1480	TC5 mode register (TCMODE5)			36.11.1.1/-8
0x9_1482	TC5 cell delineation state machine register (CDSMR5)			36.11.1.2/-10
0x9_1484	TC5 event register (TCER5)			36.11.1.3/-10
0x9_1486	TC5 received cells counter (TC_RCC5)			36.11.3.1/-14
0x9_1488	TC5 mask register (TCMR5)			36.11.1.4/-11
0x9_148A	TC5 filtered cells counter (TC_FCC5)			36.11.3.6/-14
0x9_148C	TC5 corrected cells counter (TC_CCC5)			36.11.3.4/-14
0x9_148E	TC5 idle cells counter (TC_ICC5)			36.11.3.5/-14
0x9_1490	TC5 transmitted cells counter (TC_TCC5)			36.11.3.2/-14
0x9_1492	TC5 error cells counter (TC_ECC5)			36.11.3.3/-14
0x9_1494	Reserved	—	—	—
TC Layer 6				
0x9_14A0	TC6 mode register (TCMODE6)			36.11.1.1/-8
0x9_14A2	TC6 cell delineation state machine register (CDSMR6)			36.11.1.2/-10
0x9_14A4	TC6 event register (TCER6)			36.11.1.3/-10
0x9_14A6	TC6 received cells counter (TC_RCC6)			36.11.3.1/-14
0x9_14A8	TC6 mask register (TCMR6)			36.11.1.4/-11
0x9_14AA	TC6 filtered cells counter (TC_FCC6)			36.11.3.6/-14
0x9_14AC	TC6 corrected cells counter (TC_CCC6)			36.11.3.4/-14

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_14AE	TC6 idle cells counter (TC_ICC6)			36.11.3.5/-14
0x9_14B0	TC6 transmitted cells counter (TC_TCC6)			36.11.3.2/-14
0x9_14B2	TC6 error cells counter (TC_ECC6)			36.11.3.3/-14
0x9_14B4	Reserved	—	—	—
TC Layer 7				
0x9_14C0	TC7 mode register (TCMODE7)			36.11.1.1/-8
0x9_14C2	TC7 cell delineation state machine register (CDSMR7)			36.11.1.2/-10
0x9_14C4	TC7 event register (TCER7)			36.11.1.3/-10
0x9_14C6	TC7 received cells counter (TC_RCC7)			36.11.3.1/-14
0x9_14C8	TC7 mask register (TCMR7)			36.11.1.4/-11
0x9_14CA	TC7 filtered cells counter (TC_FCC7)			36.11.3.6/-14
0x9_14CC	TC7 corrected cells counter (TC_CCC7)			36.11.3.4/-14
0x9_14CE	TC7 idle cells counter (TC_ICC7)			36.11.3.5/-14
0x9_14D0	TC7 transmitted cells counter (TC_TCC7)			36.11.3.2/-14
0x9_14D2	TC7 error cells counter (TC_ECC7)			36.11.3.3/-14
0x9_14D4	Reserved	—	—	—
TC Layer 8				
0x9_14E0	TC8 mode register (TCMODE8)			36.11.1.1/-8
0x9_14E2	TC8 cell delineation state machine register (CDSMR8)			36.11.1.2/-10
0x9_14E4	TC8 event register (TCER8)			36.11.1.3/-10
0x9_14E6	TC8 received cells counter (TC_RCC8)			36.11.3.1/-14
0x9_14E8	TC8 mask register (TCMR8)			36.11.1.4/-11
0x9_14EA	TC8 filtered cells counter (TC_FCC8)			36.11.3.6/-14
0x9_14EC	TC8 corrected cells counter (TC_CCC8)			36.11.3.4/-14
0x9_14EE	TC8 idle cells counter (TC_ICC8)			36.11.3.5/-14
0x9_14F0	TC8 transmitted cells counter (TC_TCC8)			36.11.3.2/-14
0x9_14F2	TC8 error cells counter (TC_ECC8)			36.11.3.3/-14
0x9_14F4	Reserved	—	—	—
TC Layer—General				
0x9_1500	TC general status register (TCGSR)			36.11.2.2/-13
0x9_1502	TC general event register (TCGER)			36.11.2.1/-12
BRGs 5–8				

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_15F0	BRG5 configuration register (BRGC5)	R/W	0x0000_0000	24.2/-4
0x9_15F4	BRG6 configuration register (BRGC6)	R/W	0x0000_0000	
0x9_15F8	BRG7 configuration register (BRGC7)	R/W	0x0000_0000	
0x9_15FC	BRG8 configuration register (BRGC8)	R/W	0x0000_0000	
0x9_1600–0x9_185F	Reserved	—	—	—
I^2C				
0x9_1860	I^2C mode register (I2MOD)	R/W	0x00	41.4.1/-6
0x9_1861	Reserved	—	—	—
0x9_1864	I^2C address register (I2ADD)	R/W	0x00	41.4.2/-7
0x9_1865	Reserved	—	—	—
0x9_1868	I^2C BRG register (I2BRG)	R/W	0x00	41.4.3/-7
0x9_1869	Reserved	—	—	—
0x9_186C	I^2C command register (I2COM)	R/W	0x00	41.4.5/-8
0x9_186D	Reserved	—	—	—
0x9_1870	I^2C event register (I2CER)	R/W	0x00	41.4.4/-8
0x9_1871	Reserved	—	—	—
0x9_1874	I^2C mask register (I2CMR)	R/W	0x00	41.4.4/-8
0x9_1875–0x9_19BF	Reserved	—	—	—
Communications Processor				
0x9_19C0	Communications processor command register (CPCR)	R/W	0x0000_0000	20.4.1/-27
0x9_19C4	CP configuration register (RCCR)	R/W	0x0000_0000	20.3.6/-24
0x9_19C8–0x9_19D5	Reserved	—	—	—
0x9_19D6	CP timers event register (RTER)	R/W	0x0000	20.6.4/-39
0x9_19DA	CP timers mask register (RTMR)	R/W	0x0000	
0x9_19DC	CP time-stamp timer control register (RTSCR)	R/W	0x0000	20.3.7/-25
0x9_19DE	Reserved	—	—	—
0x9_19E0	CP time-stamp register (RTSR)	R/W	0x0000_0000	20.3.8/-26

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
BRGs 1–4				
0x9_19F0	BRG1 configuration register (BRGC1)	R/W	0x0000_0000	24.2/-4
0x9_19F4	BRG2 configuration register (BRGC2)	R/W	0x0000_0000	
0x9_19F8	BRG3 configuration register (BRGC3)	R/W	0x0000_0000	
0x9_19FC	BRG4 configuration register (BRGC4)	R/W	0x0000_0000	
SCC1				
0x9_1A00	SCC1 general mode register (GSMR_L1)	R/W	0x0000_0000	27.2/-3
0x9_1A04	SCC1 general mode register (GSMR_H1)	R/W	0x0000_0000	
0x9_1A08	SCC1 protocol-specific mode register (PSMR1)	R/W	0x0000	27.2.1/-9 28.16/-14 (UART) 29.8/-7 (HDLC) 30.11/-10 (BISYNC) 31.9/-9 (Transparent)
0x9_1A0A	Reserved	—	—	—
0x9_1A0C	SCC1 transmit-on-demand register (TODR1)	R/W	0x0000	27.2.3/-10
0x9_1A0E	SCC1 data synchronization register (DSR1)	R/W	0x7E7E	27.2.2/-9
0x9_1A10	SCC1 event register (SCCE1)	R/W		28.19/-20 (UART) 29.11/-13 (HDLC) 30.14/-15 (BISYNC) 31.12/-12 (Transparent)
0x9_1A14	SCC1 mask register (SCCM1)	R/W	0x0000	
0x9_1A16	Reserved	—	—	—
0x9_1A17	SCC1 status register (SCCS1)	R/W	0x00	28.20/-22 (UART) 29.12/-15 (HDLC) 30.15/-16 (BISYNC) 31.13/-13 (Transparent)
0x9_1A18– 0x9_1A1F	Reserved	—	—	—
SCC2				
0x9_1A20	SCC2 general mode register (low) (GSMR_L2)	R/W	0x0000_0000	27.2/-3
0x9_1A24	SCC2 general mode register (high) (GSMR_H2)	R/W	0x0000_0000	

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_1A28	SCC2 protocol-specific mode register (PSMR2)	R/W	0x0000	27.2.1/-9 28.16/-14 (UART) 29.8/-7 (HDLC) 30.11/-10 (BISYNC) 31.9/-9 (Transparent)
0x9_1A2A	Reserved	—	—	—
0x9_1A2C	SCC2 transmit-on-demand register (TODR2)	R/W	0x0000	27.2.3/-10
0x9_1A2E	SCC2 data synchronization register (DSR2)	R/W	0x7E7E	27.2.2/-9
0x9_1A30	SCC2 event register (SCCE2)	R/W		28.19/-20 (UART) 29.11/-13 (HDLC) 30.14/-15 (BISYNC) 31.12/-12 (Transparent)
0x9_1A34	SCC2 mask register (SCCM2)	R/W	0x0000	
0x9_1A36	Reserved	—	—	—
0x9_1A37	SCC2 status register (SCCS2)	R/W	0x00	28.20/-22 (UART) 29.12/-15 (HDLC) 30.15/-16 (BISYNC) 31.13/-13 (Transparent)
0x9_1A38–0x9_1A3F	Reserved	—	—	—
SCC3				
0x9_1A40	SCC3 general mode register (GSMR_L3)	R/W	0x0000_0000	27.2/-3
0x9_1A44	SCC3 general mode register (GSMR_H3)	R/W	0x0000_0000	
0x9_1A48	SCC3 protocol-specific mode register (PSMR3)	R/W	0x0000	27.2.1/-9 28.16/-14 (UART) 29.8/-7 (HDLC) 30.11/-10 (BISYNC) 31.9/-9 (Transparent)
0x9_1A4A	Reserved	—	—	—
0x9_1A4C	SCC3 transmit on demand register (TODR3)	R/W	0x0000	27.2.3/-10
0x9_1A4E	SCC3 data synchronization register (DSR3)	R/W	0x7E7E	27.2.2/-9

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_1A50	SCC3 event register (SCCE3)	R/W		28.19/-20 (UART) 29.11/-13 (HDLC) 30.14/-15 (BISYNC) 31.12/-12 (Transparent)
0x9_1A54	SCC3 mask register (SCCM3)	R/W	0x0000	
0x9_1A56	Reserved	—	—	—
0x9_1A57	SCC3 status register (SCCS3)	R/W	0x00	28.20/-22 (UART) 29.12/-15 (HDLC) 30.15/-16 (BISYNC) 31.13/-13 (Transparent)
0x9_1A58–0x9_1A5F	Reserved	—	—	—
SCC4				
0x9_1A60	SCC4 general mode register (GSMR_L4)	R/W	0x0000_0000	27.2/-3
0x9_1A64	SCC4 general mode register (GSMR_H4)	R/W	0x0000_0000	
0x9_1A68	SCC4 protocol-specific mode register (PSMR4)	R/W	0x0000	27.2.1/-9 28.16/-14 (UART) 29.8/-7 (HDLC) 30.11/-10 (BISYNC) 31.9/-9 (Transparent)
0x9_1A6A	Reserved	—	—	—
0x9_1A6C	SCC4 transmit on-demand register (TODR4)	R/W	0x0000	27.2.3/-10
0x9_1A6E	SCC4 data synchronization register (DSR4)	R/W	0x7E7E	27.2.2/-9
0x9_1A70	SCC4 event register (SCCE4)	R/W		28.19/-20 (UART) 29.11/-13 (HDLC) 30.14/-15 (BISYNC) 31.12/-12 (Transparent)
0x9_1A74	SCC4 mask register (SCCM4)	R/W	0x0000	
0x9_1A76	Reserved	—	—	—

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_1A77	SCC4 status register (SCCS4)	—	—	28.20/-22 (UART) 29.12/-15 (HDLC) 30.15/-16 (BISYNC) 31.13/-13 (Transparent)
0x9_1A78–0x9_1A7F	Reserved	—	—	—
SPI				
0x9_1AA0	SPI mode register (SPMODE)	R/W	0x0000	40.4.1/-7
0x9_1AA2	Reserved	—	—	—
0x9_1AA6	SPI event register (SPIE)	R/W	0x00	40.4.2/-10
0x9_1AA7	Reserved	—	—	—
0x9_1AAA	SPI mask register (SPIM)	R/W	0x00	40.4.2/-10
0x9_1AAB	Reserved	—	—	—
0x9_1AAD	SPI command register (SPCOM)	W	0x00	40.4.3/-11
0x9_1AA7–0x9_1AFF	Reserved	—	—	—
CPM Mux				
0x9_1B00	CPM mux SI1 clock route register (CMXSI1CR)	R/W	0x00	23.4.2/-12
0x9_1B02	CPM mux SI2 clock route register (CMXSI2CR)	R/W	0x00	23.4.3/-12
0x9_1B03	Reserved	—	—	—
0x9_1B04	CPM mux FCC clock route register (CMXFCR)	R/W	0x0000_0000	23.4.4/-13
0x9_1B08	CPM mux SCC clock route register (CMXSCR)	R/W	0x0000_0000	23.4.5/-16
0x9_1B0C	Reserved	—	—	—
0x9_1B0E	CPM mux UTOPIA address register (CMXUAR)	R/W	0x0000	23.4.1/-8
0x9_1B10–0x9_1B1F	Reserved	—	—	—
SI1 Registers				
0x9_1B20	SI1 TDMA1 mode register (SI1AMR)	R/W	0x0000	22.5.2/-18
0x9_1B22	SI1 TDMB1 mode register (SI1BMR)	R/W	0x0000	
0x9_1B24	SI1 TDMC1 mode register (SI1CMR)	R/W	0x0000	
0x9_1B26	SI1 TDMD1 mode register (SI1DMR)	R/W	0x0000	
0x9_1B28	SI1 global mode register (SI1GMR)	R/W	0x00	22.5.1/-18

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_1B29	Reserved	—	—	—
0x9_1B2A	SI1 command register (SI1CMDR)	R/W	0x00	22.5.4/-25
0x9_1B2B	Reserved	—	—	—
0x9_1B2C	SI1 status register (SI1STR)	R/W	0x00	22.5.5/-26
0x9_1B2D	Reserved	—	—	—
0x9_1B2E	SI1 RAM shadow address register (SI1RSR)	R/W	0x0000	22.5.3/-24
MCC1 Registers				
0x9_1B30	MCC1 event register (MCCE1)	R/W	0x0000	33.10.1/-18
0x9_1B32	Reserved	—	—	—
0x9_1B34	MCC1 mask register (MCCM1)	R/W	0x0000	33.10.1/-18
0x9_1B36	Reserved	—	—	—
0x9_1B38	MCC1 configuration register (MCCF1)	R/W	0x00	33.8/-15
0x9_1B39–0x9_1B3F	Reserved	—	—	—
SI2 Registers				
0x9_1B40	SI2 TDMA2 mode register (SI2AMR)	R/W	0x0000	22.5.2/-18
0x9_1B42	SI2 TDMB2 mode register (SI2BMR)	R/W	0x0000	
0x9_1B44	SI2 TDMC2 mode register (SI2CMR)	R/W	0x0000	
0x9_1B46	SI2 TDMD2 mode register (SI2DMR)	R/W	0x0000	
0x9_1B48	SI2 global mode register (SI2GMR)	R/W	0x00	22.5.1/-18
0x9_1B49	Reserved	—	—	—
0x9_1B4A	SI2 command register (SI2CMDR)	R/W	0x00	22.5.4/-25
0x9_1B4B	Reserved	—	—	—
0x9_1B4C	SI2 status register (SI2STR)	R/W	0x00	22.5.5/-26
0x9_1B4D	Reserved	—	—	—
0x9_1B4E	SI2 RAM shadow address register (SI2RSR)	R/W	0x0000	22.5.3/-24
MCC2 Registers				
0x9_1B50	MCC2 event register (MCCE2)	R/W	0x0000	33.10.1/-18
0x9_1B52	Reserved	—	—	—
0x9_1B54	MCC2 mask register (MCCM2)	R/W	0x0000	33.10.1/-18
0x9_1B56	Reserved	—	—	—
0x9_1B58	MCC2 configuration register (MCCF2)	R/W	0x00	33.8/-15

Table 20-1. MPC8560 Internal Memory Map

Address (offset)	Register	Access	Reset	Section/Page
0x9_1B59–0x9_1FFF	Reserved	—	—	—
SI1 RAM				
0x9_2000–0x9_21FF	SI 1 transmit routing RAM (SI1TxRAM)			22.4.3/-10
0x9_2200–0x9_23FF	Reserved	—	—	—
0x9_2400–0x9_25FF	SI 1 receive routing RAM (SI1RxRAM)			22.4.3/-10
0x9_2600–0x9_27FF	Reserved	—	—	—
SI2 RAM				
0x9_2800–0x9_29FF	SI 2 transmit routing RAM (SI2TxRAM			22.4.3/-10
0x9_2A00–0x9_2BFF	Reserved	—	—	—
0x9_2C00–0x9_2DFF	SI 2 receive routing RAM (SI2RxRAM)			22.4.3/-10
0x9_2E00–0x9_3FFF	Reserved	—	—	—
CPM Registers				
0xA_0000–0xA_7FFF	Dual-port RAM (instruction RAM only)			
0xA_8000–0xB_FFFF	Reserved	—	—	—

20.2 MPC8560 Serial Configurations

The MPC8560 offers a flexible set of communications capabilities. A subset of the possible configurations using an MPC8650 is shown in Table 20-2.

Table 20-2. Possible MPC8560 Applications

Application	MCC1	MCC2	FCC1	FCC2	FCC3	SCC1	SCC2	SCC3	SCC4
ISDN router	4 E1	4 E1	FEnet or ATM	FEnet		UART	UART	UART	UART
ATM switch	—	—	ATM	FEnet		UART	—	—	—

Table 20-2. Possible MPC8560 Applications

Application	MCC1	MCC2	FCC1	FCC2	FCC3	SCC1	SCC2	SCC3	SCC4
ATM access	—	—	ATM	FEnet	FEnet	—	—	—	—
	E3 or E1s	E3 or E1s	ATM	—	—	UART	—	—	—
GSM mobile switching center	E1s	—	FEnet or ATM Backbone	10 M HDLC	10 M HDLC	—	—	—	—

20.3 Communications Processor (CP)

The communications processor (CP), also called the RISC microcontroller, is a 32-bit controller for the CPM that resides on a separate bus from the core and, therefore, can perform tasks independent of the e500 core. The CP handles lower-layer communications tasks and DMA control, freeing the core to handle higher-layer activities. The CP works with the peripheral controllers and parallel port to implement user-programmable protocols and manage the serial DMA (SDMA) channels that transfer data between the I/O channels and memory. It also contains an internal timer used to implement up to 16 additional software timers.

The CP's architecture and instruction set are optimized for data communications and data processing required by many wire-line and wireless communications standards.

20.3.1 Features

The following is a list of the CP's primary features.

- One system clock cycle per instruction
- 32-bit instruction object code
- Executes code from internal ROM or instruction RAM
- 32-bit ALU data path
- 64-bit internal RAM access
- Optimized for communications processing
- Performs DMA bursting of serial data from/to internal RAM/external memory
- Tuned for communications environments—instruction set supports CRC computation, and bit manipulation
- Internal timer
- Interfaces with the CPU through 32 Kbytes of internal dual-port RAM and virtual DMA channels for each serial channel
- Handles serial protocols

20.3.2 CP Block Diagram

The CP contains the following functional units:

- Scheduler and sequencer
- Instruction decoder
- Execution unit
- Load/store unit (LSU)
- Block transfer module (BTM)—moves data between serial FIFO and RAM
- Eight general purpose registers (GPRs)
- Special registers, CRC machine, HDLC framer

The CP also gives SDMA commands to the SDMA. The CP interfaces with the dual-port RAM for loading and storing data.

Figure 20-2 shows the CP block diagram.

20.3.3 e500 Core Interface

The CP communicates with the e500 core in several ways:

- Many parameters are exchanged through the dual-port RAM.
- The CP can execute special commands issued by the core. These commands should only be issued in special situations like exceptions or error recovery.
- The CP generates interrupts to the CPM interrupt controller, which is connected to the programmable interrupt controller (PIC) of the MPC8560.
- The e500 core can read the CPM status/event registers at any time.

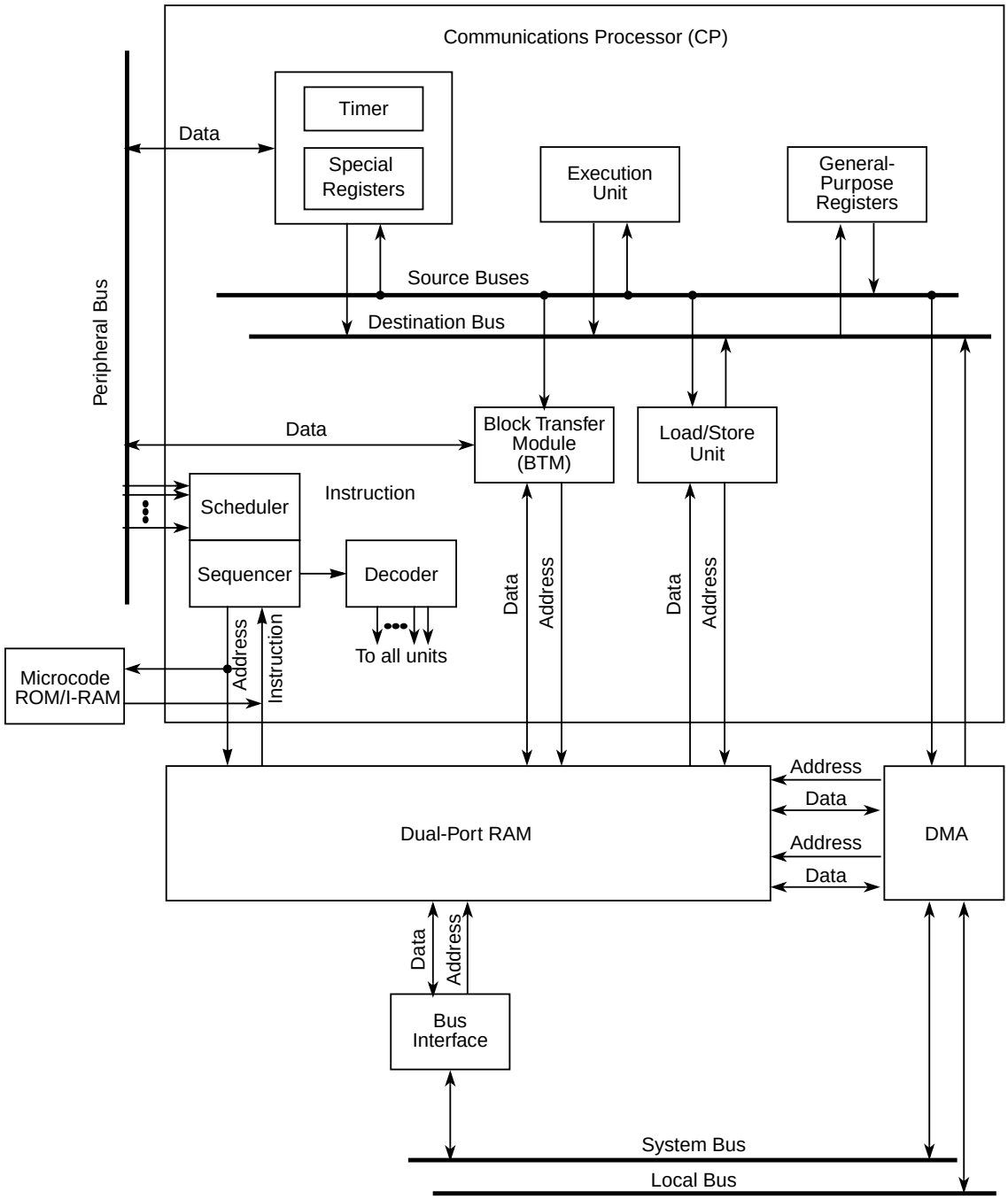


Figure 20-2. Communications Processor (CP) Block Diagram

20.3.4 Peripheral Interface

The CP uses the peripheral bus to communicate with all of its peripherals. Each FCC and each SCC has a separate receive and transmit FIFOs. The FCC FIFOs are 192 bytes. The

SCC FIFOs are 32 bytes. The SPI and I^2C are all double-buffered, creating effective FIFO sizes of two characters.

Table 20-3 shows the order in which the CP handles requests from peripherals from highest to lowest priority.

Table 20-3. Peripheral Prioritization

Priority	Request
1	Reset in the CPCR or $\overline{\text{SRESET}}$
2	SDMA bus error
3	Commands issued to the CPCR
4	Emergency (from FCCs, MCCs, and SCCs)
5	Reserved
6	FCC1 receive
7	FCC1 transmit
8	MCC1 receive
9	MCC2 receive
10	MCC1 transmit
11	MCC2 transmit
12	FCC2 receive
13	FCC2 transmit
14	FCC3 receive
15	FCC3 transmit
16	SCC1 receive
17	SCC1 transmit
18	SCC2 receive
19	SCC2 transmit
20	SCC3 receive
21	SCC3 transmit
22	SCC4 receive
23	SCC4 transmit
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	SPI receive

Table 20-3. Peripheral Prioritization (continued)

Priority	Request
30	SPI transmit
31	I^2C receive
32	I^2C transmit
33	RISC timer table
34	Reserved

20.3.5 Execution from RAM

The CP has an option to execute microcode from a portion of user RAM located in the instruction RAM. In this mode, the CP fetches instructions from both the instruction RAM and its own private ROM. This mode allows Motorola to add new protocols or enhancements to the MPC8560 in the form of RAM microcode packages. If preferred, the user can obtain binary microcode from Motorola and load it into the instruction RAM.

20.3.6 RISC Controller Configuration Register (RCCR)

The RISC controller configuration register (RCCR), shown in Figure 20-3, configures the CP to run microcode from ROM or instruction RAM and controls the CP's internal timer. The register is cleared at reset.

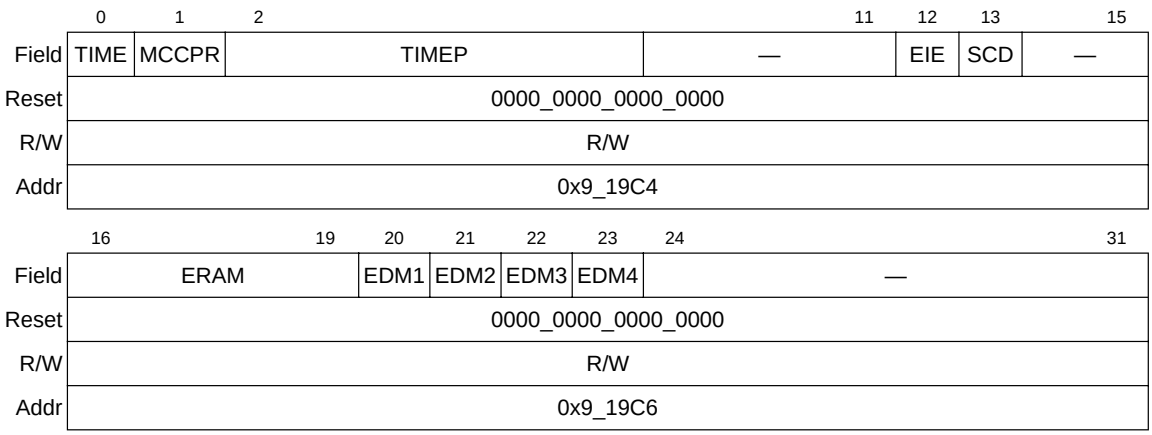


Figure 20-3. RISC Controller Configuration Register (RCCR)

RCCR bit fields are described in Table 20-4.

Table 20-4. RISC Controller Configuration Register Field Descriptions

Bits	Name	Description
0	TIME	Timer enable. Enables the CP internal timer that generates a tick to the CP based on the value programmed into the TIMEP field. TIME can be modified at any time to start or stop the scanning of the RISC timer tables.
1	MCCPR	MCC request priority. Controls the priority of the MCCs in relation to the other communication peripherals. See Table 20-3 for more information. 0 Original CPM priority scheme. MCCx priority behaves according to Table 20-3. 1 MCC priority remains at emergency level, priority level 4.
2–7	TIMEP	Timer period controls the CP timer tick. The RISC timer tables are scanned on each timer tick and the input to the timer tick generator is the general system clock (133/166 MHz) divided by 1024. The formula is (TIMEP + 1) × 1024 = (general system clock period). Thus, a value of 0 stored in these bits gives a timer tick of 1 × (1024) = 1024 general system clocks and a value of 63 (decimal) gives a timer tick of 64 × (1024) = 65,536 general system clocks.
8–11	—	Reserved, should be cleared.
12	EIE	External interrupt enable. When EIE is set, DREQ1 acts as an external interrupt to the CP. Configure as instructed in the download process of a Motorola-supplied RAM microcode package. 0 DREQ1 cannot interrupt the CP. 1 DREQ1 will interrupt the CP.
13	—	Reserved.
14–15	—	Reserved, should be cleared.
16–19	ERAM	Enable RAM microcode. Configure as instructed in the download process of a Motorola-supplied RAM microcode package. Otherwise, it should not be used. 0000 Disable microcode program execution from the internal RAM. 0100 Microcode is executed from the Instruction RAM. Other combinations of these bits are not valid and must not be used.
20–23	EDM*x*	Edge detect mode. DREQ*x* asserts as follows: 0 Low-to-high change 1 High-to-low change
24–31	—	Reserved, should be cleared.

20.3.7 RISC Time-Stamp Control Register (RTSCR)

The RISC time-stamp control register (RTSCR), shown in Figure 20-4, configures the RISC time-stamp timer (RTSR). The time-stamp timer is used by the ATM and the HDLC controllers. For application examples, see Section 35.5.3, “ABR Flow Control Setup,” and Section 38.6, “HDLC Mode Register (FPSMR).”

	0–4	5	6–15
Field	—	RTE	RTPS (Timer Prescale)
Reset	0000_0000_0000_0000		
R/W	R/W		
Addr	0x9_19DC		

Figure 20-4. RISC Time-Stamp Control Register (RTSCR)

Table 20-5 describes RTSCR fields.

Table 20-5. RTSCR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	RTE	Time stamp enable. 0 Disable time-stamp timer 1 Enable time-stamp timer
6–15	RTPS	Time-stamp timer pre-scale. Must be programmed to generate a 1-µs period input clock to the time-stamp timer. (Time-stamp frequency = (CPM frequency)/(RTPS + 2)

20.3.8 RISC Time-Stamp Register (RTSR)

The RISC time-stamp register (RTSR), shown in Figure 20-5, contains the time stamp.

	0–15
Field	Time Stamp
Reset	—
R/W	R
Addr	0x9_19E0

	16–31
Field	Time Stamp
Reset	—
R/W	R
Addr	0x9_19E2

Figure 20-5. . RISC Time-Stamp Register (RTSR)

After reset, setting RTSCR[RTE] causes the time stamp to start counting microseconds from zero.

20.3.9 RISC Microcode Revision Number

The CP writes a revision number stored in its ROM to an dual-port RAM location called REV_NUM that resides in the miscellaneous parameter RAM. The other locations are reserved for future use.

Table 20-6. RISC Microcode Revision Number

Address	Name	Width	Description
RAM Base + 0x8AF0	REV_NUM	Hword	Microcode revision number. TBD
RAM Base + 0x8AF2	—	Hword	Reserved

20.4 Command Set

The core issues commands to the CP by writing to the CP command register (CPCR). The CPCR rarely needs to be accessed. For example, to terminate the transmission of an SCC's frame without waiting until the end, a STOP TX command must be issued through the CP command register (CPCR).

20.4.1 CP Command Register (CPCR)

The core should set CPCR[FLG], shown in Figure 20-6, when it issues a command and the CP clears FLG after completing the command, thus indicating to the core that it is ready for the next command. Subsequent commands to the CPCR can be given only after FLG is clear. However, the software reset command issued by setting RST does not depend on the state of FLG, but the core should still set FLG when setting RST.

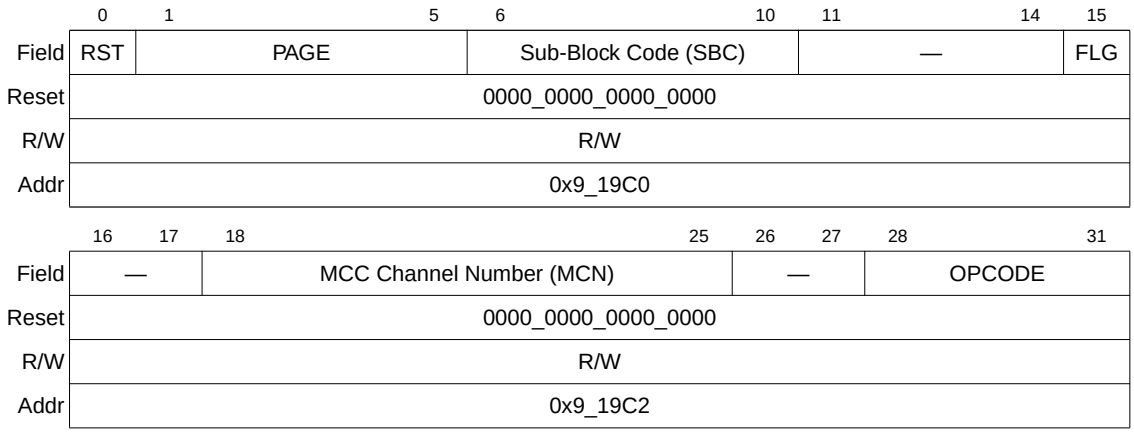


Figure 20-6. CP Command Register (CPCR)

Table 20-7 describes CPCR fields.

Table 20-7. CP Command Register Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>RST</td><td>Software reset command. Set by the core and cleared by the CP. When this command is executed, RST and FLG bit are cleared within two general system clocks. The CPM reset routine is approximately 60 clocks long, but the user can begin initialization of the CPM immediately after this command is issued.
RST is useful when the core wants to reset the registers and parameters for all the channels (FCCs, SCCs, SPI, I²C, MCC) as well as the CP and RISC timer tables. However, this command does not affect the serial interface (SIx) or parallel I/O registers.</td></tr>
<tr><td>1–5</td><td>PAGE</td><td>Indicates the parameter RAM page number associated with the sub-block being served. See the SBC description for page numbers.</td></tr>
<tr><td>6–10</td><td>SBC</td><td>Sub-block code. Set by the core to specify the sub-block on which the command is to operate. Set according to OPCODE[28–31]. Refer to Table 13-7.
<table>
<tr><th>SubBlock</th><th>Code</th><th>Page</th><th>Sub-Block</th><th>Code</th><th>Page</th></tr>
<tr><td>FCC1 [1]</td><td>10000
(for ATM: 01110)</td><td>00100</td><td>SPI</td><td>01010</td><td>01001</td></tr>
<tr><td>FCC2[1]</td><td>10001
(for ATM: 01110)</td><td>00101</td><td>I²C</td><td>01011</td><td>01010</td></tr>
<tr><td>FCC3</td><td>10010</td><td>00110</td><td>Timer</td><td>01111</td><td>01010</td></tr>
<tr><td>SCC1</td><td>00100</td><td>00000</td><td>MCC1</td><td>11100</td><td>00111</td></tr>
<tr><td>SCC2</td><td>00101</td><td>00001</td><td>MCC2</td><td>11101</td><td>01000</td></tr>
<tr><td>SCC3</td><td>00110</td><td>00010</td><td>Reserved</td><td>10100</td><td>00111</td></tr>
<tr><td>SCC4</td><td>00111</td><td>00011</td><td>Reserved</td><td>10101</td><td>01000</td></tr>
<tr><td>Reserved</td><td>01000</td><td>00111</td><td>Reserved</td><td>10110</td><td>01001</td></tr>
<tr><td>Reserved</td><td>01001</td><td>01000</td><td>Reserved</td><td>10111</td><td>01010</td></tr>
<tr><td>RAND</td><td>01110</td><td>01010</td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td>11–14</td><td>—</td><td>Reserved</td></tr>
<tr><td>15</td><td>FLG</td><td>Command semaphore flag. Set by the core and cleared by the CP.
0 The CP is ready to receive a new command.
1 The CPCR contains a command that the CP is currently processing. The CP clears this bit at the end of command execution or after reset.</td></tr>
<tr><td>16–17</td><td>—</td><td>Reserved</td></tr>
<tr><td>18–25</td><td>MCN</td><td>MCC channel number. Specifies the channel number in the case of an MCC command.
In FCC protocols, this field contains the protocol code as follows:
0x00 HDLC
0x0A ATM
0x0F Transparent</td></tr>
<tr><td>26–27</td><td>—</td><td>Reserved</td></tr>
<tr><td>28–31</td><td>OPCODE</td><td>Operation code. Settings are listed in Table 20-8.</td></tr>
</table>

[1] Set according to OPCODE[28–31]. If OPCODE is 1010, SBC must be 01110. Refer to Table 20-8.

20.4.1.1 CP Commands

The CP command opcodes are shown in Table 20-8.

Table 20-8. CP Command Opcodes

Opcode	Channel						
	FCC	SCC	SPI	I^2C	MCC	Timer	Special
0000	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	INIT RX AND TX PARAMS	—	—
0001	INIT RX PARAMS	INIT RX PARAMS	INIT RX PARAMS	INIT RX PARAMS	INIT RX PARAMS	—	—
0010	INIT TX PARAMS	INIT TX PARAMS	INIT TX PARAMS	INIT TX PARAMS	INIT TX PARAMS	—	—
0011	ENTER HUNT MODE	ENTER HUNT MODE	—	—	INIT MCC RX AND TX PARAMS (ONE CHANNEL)	—	—
0100	STOP TX	STOP TX	—	—	MCC STOP TX	—	—
0101	GRACEFUL STOP TX	GRACEFUL STOP TX	—	—	INIT MCC TX PARAMS (ONE CHANNEL)	—	—
0110	RESTART TX	RESTART TX	—	—	INIT MCC RX PARAMS (ONE CHANNEL)	—	—
0111	—	—	—	—	—	—	—
1000	SET GROUP ADDRESS	SET GROUP ADDRESS	—	—	—	SET TIMER	—
1001	—	—	—	—	MCC STOP RX	—	—
1010	ATM TRANSMIT COMMAND [1]	RESET BCS	—	—	—	—	—
1011	—	—	—	—	—	—	—
1100	—	—	—	—	—	—	RANDOM NUMBER
11xx	Undefined. Reserved for use by Motorola-supplied RAM microcodes.						

[1] See FCC1 and FCC2 in SBC[6–10] in Table 20-7.

NOTE

If a reserved command is issued, the CPM enters an unknown state that requires an external reset to recover.

The commands in Table 20-8 are described in Table 20-9.

Table 20-9. Command Descriptions

Command	Description
INIT TX AND RX PARAMS	Initialize transmit and receive parameters. Initializes the transmit and receive parameters in the parameter RAM to the values that they had after the last reset of the CP. This command is especially useful when switching protocols on a given peripheral controller.
INIT MCC RX AND TX PARAMS — ONE CHANNEL [1]	Initialize receive and transmit parameters. Initializes the receive and transmit parameters of the peripheral controller. Differs from INIT RX AND TX PARAMS in that, for the MCCs, issuing INIT RX AND TX PARAMS initializes 32 consecutive channels beginning with the channel number specified in CPCR[MCN], but issuing INIT MCC RX AND TX—ONE CHANNEL initializes only the channel in the command; see Section 33.9, “MCC Commands.”
INIT RX PARAMS	Initialize receive parameters. Initializes the receive parameters of the peripheral controller. Note that for the MCCs, issuing this command initializes only 32 channels at a time; see Section 33.9, “MCC Commands.”
INIT MCC RX PARAMS— ONE CHANNEL [1]	Initialize MCC receive parameters for only a single channel according to MCC channel number field. See Section 33.9, “MCC Commands.”
INIT TX PARAMS	Initialize transmit parameters. Initializes the transmit parameters of the peripheral controller. Note that for the MCCs, issuing this command initializes only 32 channels at a time; see Section 33.9, “MCC Commands.”
init tx params— one channel [1]	Initialize MCC transmit parameters for only a single channel according to MCC channel number field. See Section 33.9, “MCC Commands.”
ENTER HUNT MODE	Enter hunt mode. Causes the receiver to stop receiving and begin looking for a new frame. The exact operation of this command may vary depending on the protocol used.
STOP TX	Stop transmission. Aborts the transmission from this channel as soon as the transmit FIFO has been emptied. It should be used in cases where transmission needs to be stopped as quickly as possible. Transmission proceeds when the RESTART command is issued.
GRACEFUL STOP TX	Graceful stop transmission. Stops the transmission from this channel as soon as the current frame has been fully transmitted from the transmit FIFO. Transmission proceeds when the RESTART command is issued and the R-bit is set in the next TxBD.
RESTART TX	Restart transmission. Once the STOP TX command has been issued, this command is used to restart transmission at the current BD.
CLOSE RXBD	Close RxBD. Causes the receiver to close the current RxBD, making the receive buffer immediately available for manipulation by the user. Reception continues using the next available BD. Can be used to access the buffer without waiting until the buffer is completely filled by the SCC.
SET TIMER	Set timer. Activates, deactivates, or reconfigures 1 of the 16 timers in the RISC timer table.
SET GROUP ADDRESS	Set group address. Sets a bit in the hash table for the Ethernet logical group address recognition function.
RESET BCS	Reset block check sequence. Used in BISYNC mode to reset the block check sequence calculation.
MCC STOP TRANSMIT	See Section 33.9, “MCC Commands.”
MCC STOP RECEIVE	See Section 33.9, “MCC Commands.”

Table 20-9. Command Descriptions (continued)

Command	Description
MCC RESET [1]	MCC reset. Provides a hard reset to the MCC FIFOs. See Section 33.9, “MCC Commands.” To use this command, software should execute the following sequence: 1. Disable the TDM by clearing the appropriate enable bit in SIxGMR[4-7] (See Table 14-4, “SIxGMR Field Descriptions.”). 2. Issue the MCC RSET command. 3. Issue the INIT RX AND TX command. 4. Reprogram the specific MCC channel, global parameters, and any BDs that need to be updated. 5. Set the appropriate enable bit in SIxGMR[4-7]. (See Table 14-4, “SIxGMR Field Descriptions.”)
ATM TRANSMIT	See Section 35.14, “ATM Transmit Command.”
RANDOM NUMBER	Generate a random number and put it in dual-port RAM; see RAND in Table 20-11..

[1] Not available on pre-revision B.3 silicon.

NOTE

The CPM accesses BDs by initiating a DMA cycle on either the 60x or local bus. If BDs are located in DPRAM, the CPM initiates a cycle on the 60x bus because DPRAM is a slave device on the 60x bus. Therefore, a system design should not plan to access the 60x bus simultaneously with DPRAM BD fetches.

20.4.2 Command Register Example

To perform a complete reset of the CP, the value 0x8001_0000 should be written to the CPCR. Following this command, the CPCR returns the value 0x0000_0000 after two clocks.

20.4.3 Command Execution Latency

The worst-case command execution latency is 200 clocks and the typical command execution latency is about 40 clocks.

20.5 Internal RAM

The CPM has 64-Kbyte of static RAM. This RAM is split into two blocks of 32 Kbytes.

- 32 Kbytes of instruction RAM to store a microcode package of up to 8K instructions
- 32 Kbytes of dual-port data RAM to store CPM-RISC parameter RAM and data structures

Figure 20-7 is a block diagram of the internal RAM.

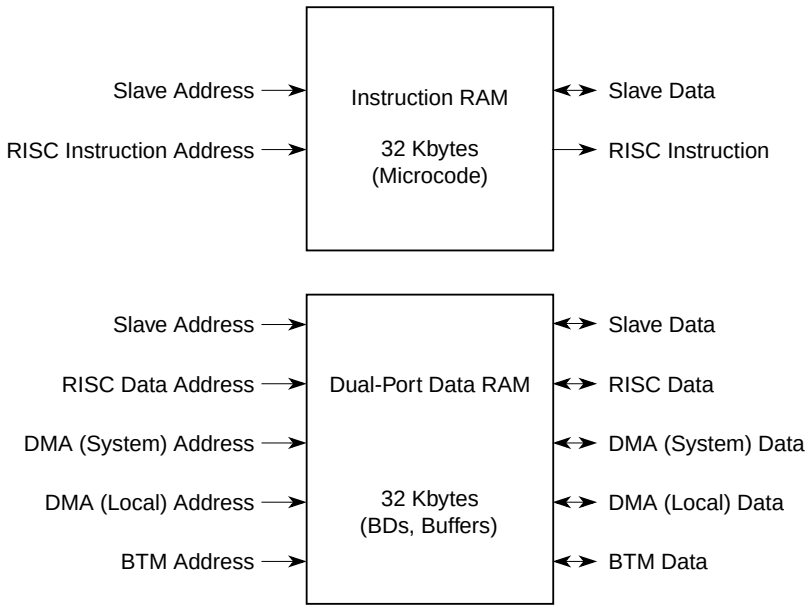


Figure 20-7. Internal RAM Block Diagram

The internal instruction RAM can be accessed by the following:

- CP instruction fetcher (in case of microcode from RAM)
- Other masters connected through the ECM (including the processor core)

The internal dual-port data RAM can be accessed by the following:

- CP load/store unit
- CP block transfer module (BTM)
- Other masters connected through the ECM (including the processor core)
- SDMA system bus
- SDMA local bus

Figure 20-8 shows the memory map of the internal instruction RAM.

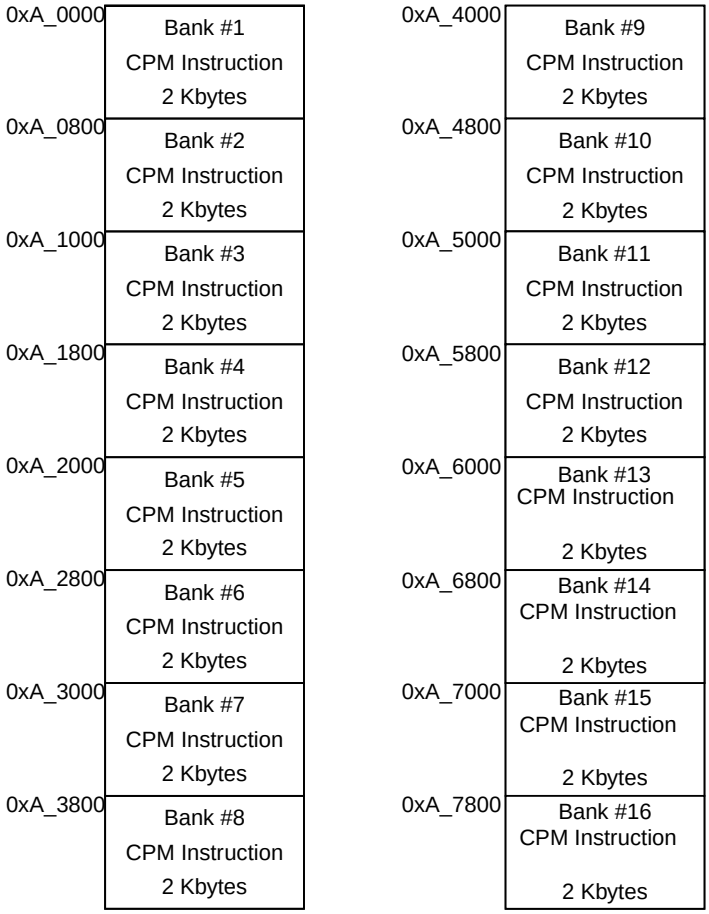


Figure 20-8. Internal Instruction RAM Memory Map

Figure 20-9 shows a memory map of the internal dual-port data RAM.

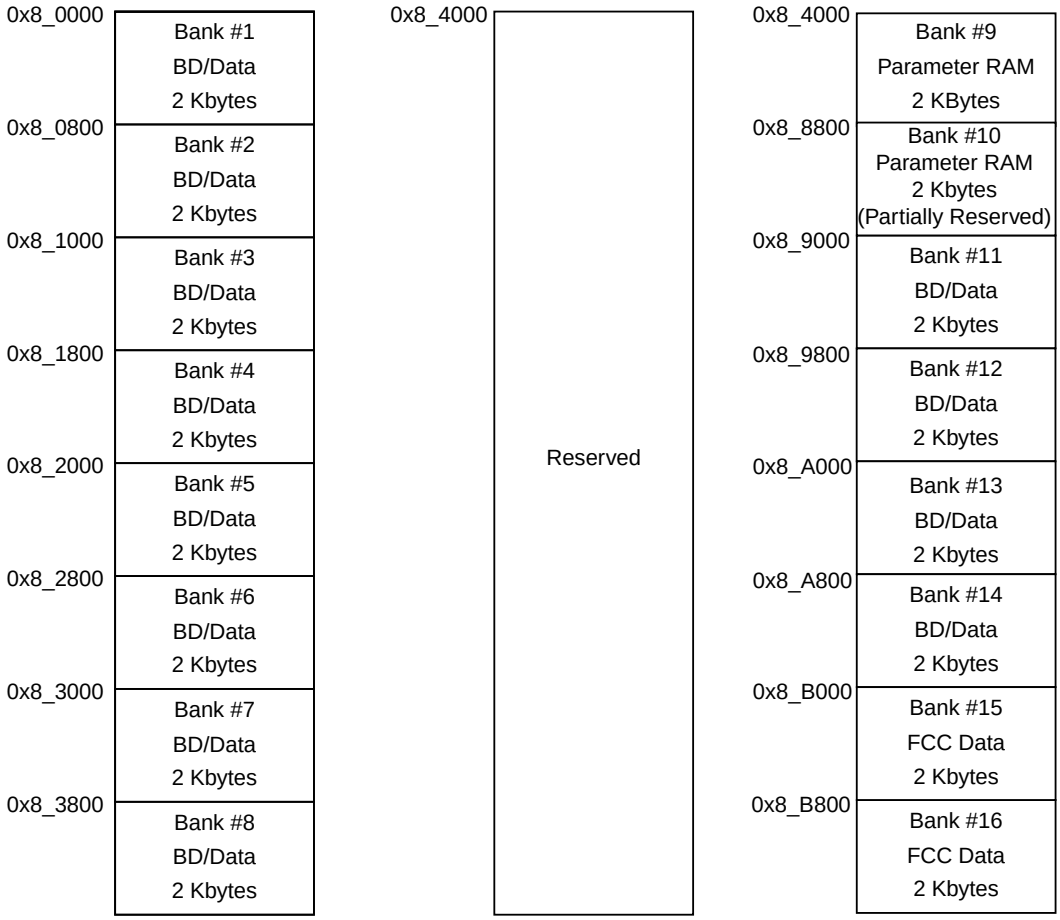


Figure 20-9. Internal Dual-Port Data RAM Memory Map

The dual-port data RAM data bus is 64-bits wide. The RAM is used for the following tasks:

- To store parameters associated with the FCCs, SCCs, MCCs, SPI, and I^2C in the parameter RAM.
- To store buffer descriptors (BDs).
- To hold data buffers (optional because data can also be stored in external memory).
- For temporary storage of FCC data moving to/from an FCC FIFO (using the BTM) from/to external memory (using SDMA).
- For additional scratch-pad RAM space for user software.

The dual-port data RAM is designed to serve multiple requests at the same cycle, as long as they are not in the same bank.

Only the parameters in the parameter RAM and the microcode RAM option require fixed addresses to be used. The BDs, buffer data and scratchpad RAM can be located in the

internal system RAM or in any unused parameter RAM, such as, in the area made available when a peripheral controller or sub-block is not being used.

Microcode can be executed from the instruction RAM (separate form the data RAM).

20.5.1 Buffer Descriptors (BDs)

The peripheral controllers (FCCs, SCCs, MCCs, SPI, and I^2C) always use BDs for controlling buffers and their BD formats are all the same, as shown in Table 20-10.

Table 20-10. Buffer Descriptor Format

Address	Descriptor
Offset + 0	Status and control
Offset + 2	Data length
Offset + 4	High-order buffer pointer
Offset + 6	Low-order buffer pointer

20.5.2 Parameter RAM

The CPM maintains a section of RAM called the parameter RAM, which contains many parameters for the operation of the FCCs, SCCs, SPI, and I^2C channels. An overview of the parameter RAM structure is shown in Table 20-11.

The exact definition of the parameter RAM is contained in each protocol subsection describing a device that uses a parameter RAM. For example, the Ethernet parameter RAM is defined differently in some locations from the HDLC-specific parameter RAM.

Table 20-11. Parameter RAM

Page	Address Offset	Peripheral	Size (Bytes)
1	0x8_8000	SCC1	256
2	0x8_8100	SCC2	256
3	0x8_8200	SCC3	256
4	0x8_8300	SCC4	256
5	0x8_8400	FCC1	256
6	0x8_8500	FCC2	256
7	0x8_8600	FCC3	256
8	0x8_8700	MCC1	128
	0x8_8780	Reserved	124
	0x8_87FC	Reserved	2
	0x8_87FE	Reserved	2
9	0x8_8800	MCC2	128
	0x8_8880	Reserved	124
	0x8_88FC	Reserved	2
	0x8_88FE	Reserved	2
10	0x8_8900	Reserved	252
	0x8_89FC	SPI_BASE	2
	0x8_89FE	Reserved	2
11	0x8_8A00	Reserved	224
	0x8_8AE0	RISC Timers	16
	0x8_8AF0	REV_NUM [1]	2
	0x8_8AF2	Reserved	2
	0x8_8AF4	Reserved	4
	0x8_8AF8	RAND	4
	0x8_8AFC	I^2C_BASE	2
	0x8_8AFE	Reserved	2
12–16	0x8_8B00	Reserved	1280

[1] Refer to Table 20-6.

20.6 RISC Timer Tables

The CP can control up to 16 software timers that are separate from the 4 general-purpose timers and the BRGs in the CPM. These timers are best used in protocols that do not require extreme precision, but in which it is preferable to free the core from scanning the software's

timer tables. These timers are clocked from an internal timer that only the CP uses. The following is a list of the RISC timer tables important features.

- Supports up to 16 timers.
- Two timer modes: one-shot and restart.
- Maskable interrupt on timer expiration.
- Programmable timer resolution as fine as 3.85 µs at 266 MHz (3.09 µs at 333 MHz).
- Maximum timeout period of 15.9 seconds at 266 MHz (12.8 seconds at 333 MHz).
- Continuously updated reference counter.

All operations on the RISC timer tables are based on a fundamental tick of the CP's internal timer that is programmed in the RCCR. The tick is a multiple of 1024 general system clocks; see Section 20.3.6, "RISC Controller Configuration Register (RCCR)."

The RISC timer tables have the lowest priority of all CP operations. Therefore, if the CP is so busy with other tasks that it does not have time to service the timer during a tick interval, one or more timer may not be updated accurately. This behavior can be used to estimate the worst-case loading of the CP; see Section 20.6.10, "Using the RISC Timers to Track CP Loading."

The timer table is configured using the RCCR, the timer table parameter RAM, and the RISC controller timer event/mask registers (RTER/RTMR), and by issuing SET TIMER to the CPCR.

20.6.1 RISC Timer Table Parameter RAM

Two areas of dual-port RAM, shown in Figure 20-10, are used for the RISC timer tables:

- The RISC timer table parameter RAM
- The RISC timer table entries

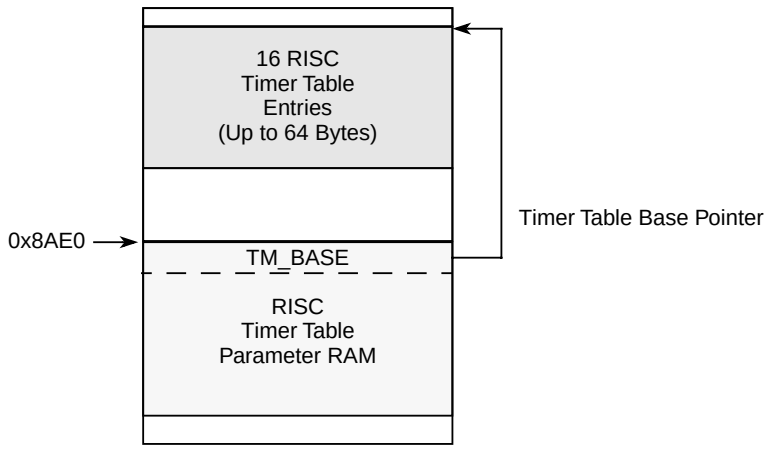


Figure 20-10. RISC Timer Table RAM Usage

The RISC timer table parameter RAM area begins at the RISC timer base address and is used for the general timer parameters; see Table 20-12.

Table 20-12. RISC Timer Table Parameter RAM

Offset[1]	Name	Description
0x00	TM_BASE	RISC timer table base address. The actual timers are a small block of memory in the dual-port RAM. TM_BASE is the offset from the beginning of the dual-port RAM where that block resides. Four bytes must be reserved at the TM_BASE for each timer used, (64 bytes if all 16 timers are used). If fewer than 16 timers are used, timers should be allocated in ascending order to save space. For example, only 8 bytes are required if two timers are needed and RISC timers 0 and 1 are enabled. TM_BASE should be word-aligned.
0x02	TM_PTR	RISC timer table pointer. This value is used exclusively by the CP to point to the next timer accessed in the timer table. It should not be modified by the user.
0x04	R_TMR	RISC timer mode register. This value is used exclusively by the CP to store the mode of the timer—one-shot (bit is 0) or restart (bit is 1). R_TMR should not be modified by the user. The SET TIMER command should be used instead.
0x06	R_TMV	RISC timer valid register. Used exclusively by the CP to determine if a timer is currently enabled. If the corresponding timer is enabled, a bit is 1. R_TMV should not be modified by the user. The SET TIMER command should be used instead.
0x08	TM_CMD	RISC timer command register. Used as a parameter location when the SET TIMER command is issued. The user should write this location before issuing the SET TIMER command. This register is defined in Section 20.6.2, “RISC Timer Command Register (TM_CMD).”
0x0C	TM_CNT	RISC timer internal count. A tick counter that the CP updates after each tick. The update occurs after the CP complete scanning the timer table.All 16 timers are scanned every tick interval regardless of whether any of them is enabled.It is updated if the CP’s internal timer is enabled, regardless of whether any of the 16 timers are enabled and it can be used to track the number of ticks the CP receives and responds to.TM_CNT is updated only after the last timer (timer 15) has been serviced. If the CP is so busy with other tasks that it does not have time to service all the timers during a tick interval, and timer 15 has not been serviced, then TM_CNT would not be updated in that tick interval.

[1] Offset from timer base address (0x8AE0).

20.6.2 RISC Timer Command Register (TM_CMD)

Figure 20-11 shows the RISC timer command register (TM_CMD).

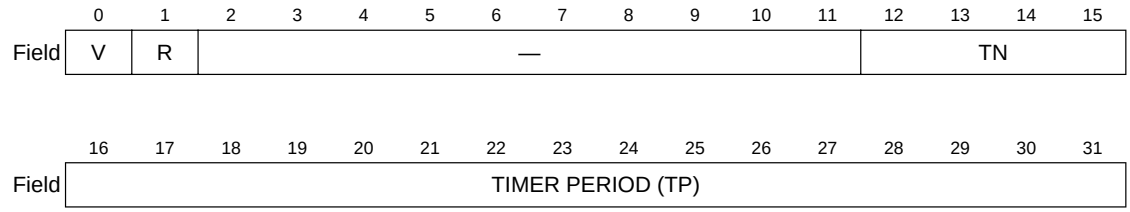


Figure 20-11. RISC Timer Command Register (TM_CMD)

TM_CMD fields are described in Table 20-13.

Table 20-13. TM_CMD Field Descriptions

Bits	Name	Description
0	V	Valid. This bit should be set to enable the timer and cleared to disable it.
1	R	Restart. Should be set for an automatic restart or cleared for a one-shot operation of the timer.
2–11	—	Reserved. These bits should be written with zeros.
12–15	TN	Timer number. A value from 0–15 signifying which timer to use—an offset into the timer table entries.
16–31	TP	Timer period. The 16-bit timeout value of the timer is zero-based. The minimum value is 1 and is programmed by writing 0x0000 to the timer period.The maximum value of the timer is 65,536 and is programmed by writing 0xFFFF.

20.6.3 RISC Timer Table Entries

The 16 timers are located in the block of memory following the TM_BASE location; each timer occupies 4 bytes. The first half-word forms the initial value of the timer written during the execution of the SET TIMER command and the next half-word is the current value of the timer that is decremented until it reaches zero. These locations should not be modified by the user. They are documented only as a debugging aid for user code. Use the SET TIMER command to initialize table values.

20.6.4 RISC Timer Event Register (RTER)/Mask Register (RTMR)

The RTER is used to report events recognized by the 16 timers and to generate interrupts. RTER can be read at any time. Bits are cleared by writing ones; writing zeros does not affect bit values.

The RISC timer mask register (RTMR) is used to enable interrupts that can be generated in the RTER. Setting an RTMR bit enables the corresponding interrupt in the RTER; clearing a bit masks the corresponding interrupt. An interrupt is generated only if the RISC timer table bit is set in the SIU interrupt mask register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TMR 15	TMR 14	TMR 13	TMR 12	TMR 11	TMR 10	TMR 9	TMR 8	TMR 7	TMR 6	TMR 5	TMR 4	TMR 3	TMR 2	TMR 1	TMR 0
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_19D6 (RTER)/0x9_19DA (RTMR)															

Figure 20-12. RISC Timer Event Register (RTER)/Mask Register (RTMR)

20.6.5 SET TIMER Command

The SET TIMER command is used to enable, disable, and configure the 16 timers in the RISC timer table and is issued to the CPCR. This means the value 0x29E1008 should be written to CPCR. However, before writing this value, the user should program the TM_CMD fields. See Section 20.6.2, “RISC Timer Command Register (TM_CMD).”

20.6.6 RISC Timer Initialization Sequence

The following sequence initializes the RISC timers:

1. Configure RCCR to determine the preferred tick interval for the entire timer table. The TIME bit is normally set at this time but can be set later if all RISC timers need to be synchronized.
2. Determine the maximum number of timers to be located in the timer table. Configure the TM_BASE in the RISC timer table parameter RAM to point to a location in the dual-port RAM with $4 \times n$ bytes available, where n is the number of timers. If n is less than 16, use timer 0 through timer n–1 to save space.
3. Clear the TM_CNT field in the RISC timer table parameter RAM to show how many ticks elapsed since the RISC internal timer was enabled. This step is optional.
4. Clear RTER, if it is not already cleared. Write ones to clear this register.
5. Configure RTMR to enable the timers that should generate interrupts. Ones enable interrupts.
6. Set the RISC timer table bit in the SIU interrupt mask register (SIMR_L[RTT]) to generate interrupts to the system. The SIU interrupt controller may require other initialization not mentioned here.
7. Configure the TM_CMD field of the RISC timer table parameter RAM. At this point, determine whether a timer is to be enabled or disabled, one-shot or restart, and what its timeout period should be. If the timer is being disabled, the parameters (other than the timer number) are ignored.
8. Issue the SET TIMER command by writing 0x29E1_0008 to the CPCR.
9. Repeat the preceding two steps for each timer to be enabled or disabled.

20.6.7 RISC Timer Initialization Example

The following sequence initializes RISC timer 0 to generate an interrupt approximately every second using a 133-MHz general system clock:

1. Write 111111 to RCCR[TIMEP] to generate the slowest clock. This value generates a tick every 65,536 clocks, which is every 485 µs at 133 MHz.

2. Configure the TM_BASE in the RISC timer table parameter RAM to point to a location in the dual-port RAM with 4 bytes available. Assuming the beginning of dual-port RAM is available, write 0x0000 to TM_BASE.
3. (Optional) Write 0x0000 to the TM_CNT field in the RISC timer table parameter RAM to see how many ticks elapsed since the RISC internal timer was enabled.
4. Write 0xFFFF to the RTER to clear any previous events.
5. Write 0x0001 to the RTMR to enable RISC timer 0 to generate an interrupt.
6. Write 0x0002_0000 to the SIU interrupt mask register (SIMR_L) to allow the RISC timers to generate a system interrupt. Initialize the SIU interrupt configuration register.
7. Write 0xC000_080D to the TM_CMD field of the RISC timer table parameter RAM. This enables RISC timer 0 to timeout after 2,061(decimal) ticks of the timer. The timer automatically restarts after it times out.
8. Write 0x29E1_0008 to the CPCR to issue the SET TIMER command.
9. Set RCCR[TIME] to enable the RISC timer to begin operation.

20.6.8 RISC Timer Interrupt Handling

The following sequence describes what normally would occur within an interrupt handler for the RISC timer tables:

1. Once an interrupt occurs, read RTER to see which timers have caused interrupts. The RISC timer event bits are usually cleared by this time.
2. Issue additional SET TIMER commands at this time or later, as preferred. Nothing needs to be done if the timer is being automatically restarted for a repetitive interrupt.
3. Clear the RTT bit in the SIU interrupt pending register (SIPNR_L).
4. Execute the RTE instruction.

20.6.9 RISC Timer Table Scan Algorithm

The CP scans the timer table once every tick. It handles each of the 16 timers at its turn and checks for other requests with higher priority to service, before handling the next one. For each valid timer in the table, the CP decrements the count and checks for a timeout. If none occurs, the CP moves to the next timer. If a timeout occurs, the CP sets the corresponding event bit in RTER. Then the CP checks to see if the timer is to be restarted and if it is, the CP leaves the timer's valid bit set in the R_TMV location and resets the current count to the initial count. Otherwise, it clears R_TMV. Once the timer table scanning has completed, the CP updates the TM_CNT value in the RISC timer table parameter RAM and stops working on the timer tables until the next tick.

If a SET TIMER command is issued, the CP makes the appropriate modifications to the timer table and parameter RAM, but does not scan the timer table until the next tick of the internal timer. It is important to use the SET TIMER command to properly synchronize timer table modifications to the execution of the CP.

20.6.10 Using the RISC Timers to Track CP Loading

The RISC timers can be used to track CP loading. The following sequence provides a way to use the 16 RISC timers to determine if the CP ever exceeds the 96% utilization level during any tick interval. Removing the timers adds a 4% margin to the CP utilization level, but the aggressive user can use this technique to push CP performance to its limit. The user should use the standard initialization sequence and incorporate the following differences:

1. Program the tick of the RISC timers to be every 1024×16 = 16,384 system clocks.
2. Disable RISC timer interrupts, if preferred.
3. Using the SET TIMER command, initialize all 16 RISC timers to have a timer period of 0xFFFF, which equates to 65,536.
4. Program one of the four general-purpose timers to increment once every tick. The general-purpose timer should be free-running and should have a timeout of 65,536.
5. After a few hours of operation, compare the general-purpose timer to the current count of RISC timer 15. If it is more than two ticks different from the general-purpose timer, the CP has, during some tick interval, exceeded the 96% utilization level.

NOTE

General-purpose timers are up counters, but RISC timers are down counters. The user should take this under consideration when comparing timer counts.

Chapter 21
CPM Interrupt Controller

Key features of the CPM interrupt controller include the following:

- Communications processor module (CPM) interrupt sources (FCCs, SCCs, MCCs, timers, I^2C, SDMA, and SPI)
- 16 external interrupt pins (port C)
- Programmable priority between SCCs, FCCs, and MCCs
- Two priority schemes for the SCCs: grouped, spread
- Programmable highest priority request
- Unique vector number for each interrupt source

21.1 Interrupt Configuration

Figure 21-1 shows the MPC8560 interrupt structure. The interrupt controller receives interrupts from internal sources from the CPM and from external pins (port C parallel I/O pins). The CPM interrupt controller concentrates all of these interrupt inputs and outputs a single interrupt signal to the EPIC.

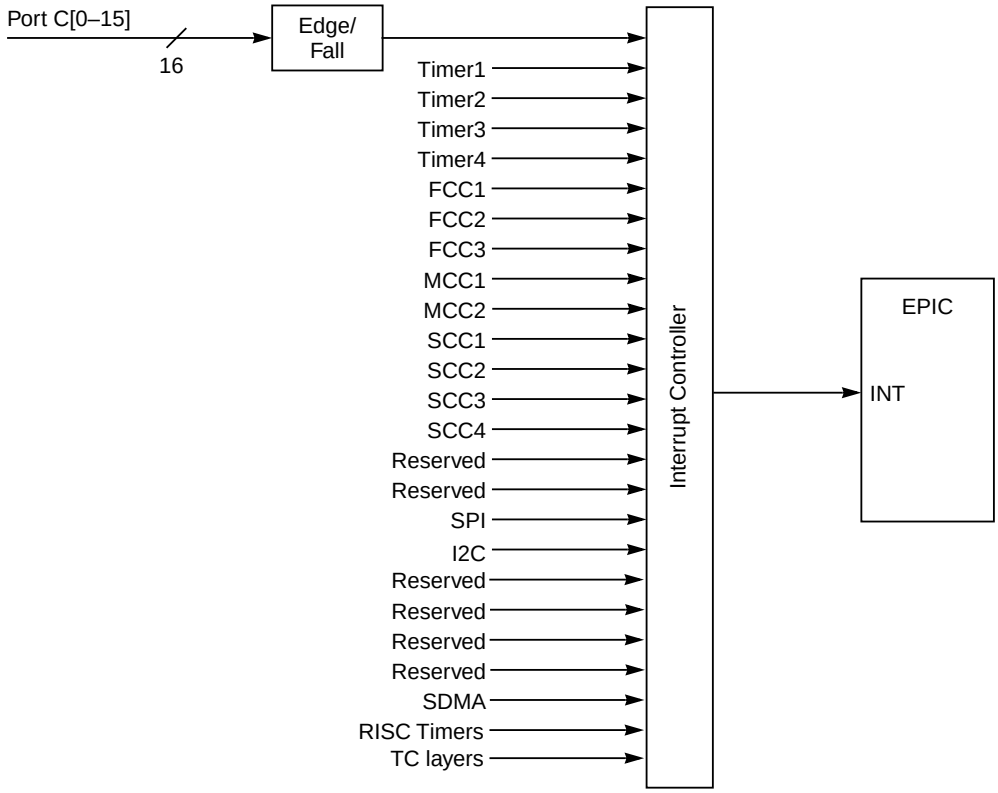


Figure 21-1. MPC8560 CPM Interrupt Structure

The CPM interrupt controller allows masking of each interrupt source. Multiple events within a CPM sub-block event are also maskable.

All interrupt sources are prioritized and bits are set in the interrupt pending register (SIPNR). On the MPC8560, the prioritization of the interrupt sources is flexible in the following two aspects:

- Relative priority of the FCCs, SCCs, and MCCs can be modified
- One interrupt source can be assigned the highest priority

When an unmasked interrupt source is pending in the SIPNR, the interrupt controller sends an interrupt request to the EPIC.

The CPM interrupt vector register (SIVEC) is updated with a 6-bit vector corresponding to the sub-block with the highest current priority.

21.2 CPM Interrupt Source Priorities

The CPM interrupt controller has 34 interrupt sources that assert one interrupt request to the EPIC. Table 21-1 shows prioritization of all interrupt sources. As described in following

sections, flexibility exists in the relative ordering of the interrupts, but, in general, relative priorities are as shown. A single interrupt priority number is associated with each table entry.

Note that the group and spread options, shown with YCC entries in Table 21-1, are described in Section 21.2.1, “SCC, FCC, and MCC Relative Priority.”

Table 21-1. Interrupt Source Priority Levels

Priority Level	Interrupt Source Description	Multiple Events
1	Highest	—
2	XCC1	Yes
3	XCC2	Yes
4	XCC3	Yes
5	XCC4	Yes
6	XCC5	Yes
7	XCC6	Yes
8	XCC7	Yes
9	XCC8	Yes
10	YCC1 (Grouped)	Yes
11	YCC2 (Grouped)	Yes
12	YCC3 (Grouped)	Yes
13	YCC4 (Grouped)	Yes
14	YCC5 (Grouped)	Yes
15	YCC6 (Grouped)	Yes
16	YCC7 (Grouped)	Yes
17	YCC8 (Grouped)	Yes
18	Parallel I/O–PC15	Yes
19	Timer 1	Yes
20	Parallel I/O–PC14	Yes
21	YCC1 (Spread)	Yes
22	Parallel I/O–PC13	Yes
23	SDMA Bus Error	Yes
24	Reserved	—
25	YCC2 (Spread)	Yes
26	Parallel I/O–PC12	No
27	Parallel I/O–PC11	No
28	Reserved	—

Table 21-1. Interrupt Source Priority Levels (continued)

Priority Level	Interrupt Source Description	Multiple Events
29	Timer 2	Yes
30	Parallel I/O–PC10	No
31	YCC3 (Spread)	Yes
32	RISC Timer Table	Yes
33	I2C	Yes
34	YCC4 (Spread)	Yes
35	Parallel I/O–PC9	No
36	Parallel I/O–PC8	No
37	Reserved	—
38	Timer 3	Yes
39	YCC5 (Spread)	Yes
40	Parallel I/O–PC7	No
41	Parallel I/O–PC6	No
42	Parallel I/O–PC5	No
43	Timer 4	Yes
44	YCC6 (Spread)	Yes
45	Parallel I/O–PC4	No
46	Reserved	—
47	SPI	Yes
48	Parallel I/O–PC3	No
49	Parallel I/O–PC2	No
50	Reserved	—
51	YCC7 (spread)	Yes
52	Reserved	—
53	Parallel I/O–PC1	No
54	Parallel I/O–PC0	No
55	YCC8(spread)	Yes
56	Reserved	—

Notice the lack of SDMA interrupt sources, which are reported through each individual FCC, SCC, SPI, or I^2C channel. The only true SDMA interrupt source is the SDMA channel bus error entry that is reported when a bus error occurs during an SDMA access. There are two ways to add flexibility to the table of CPM interrupt priorities—the FCC, MCC, and SCC relative priority option, described in Section 21.2.1, “SCC, FCC, and MCC

Relative Priority," and the highest priority option, described in Section 21.2.2, "Highest Priority Interrupt."

21.2.1 SCC, FCC, and MCC Relative Priority

The relative priority between the four SCCs, three FCCs, and MCC is programmable and can be changed dynamically. In Table 21-1 there is no entry for SCC1–SCC4, MCC1–MCC2, FCC1–FCC3, but rather there are entries for XCC1–XCC8 and YCC1–YCC8. Each SCC can be mapped to any YCC location and each FCC and MCC can be mapped to any XCC location. The SCC, FCC, and MCC priorities are programmed in the CPM interrupt priority registers (SCPRR_H and SCPRR_L) and can be changed dynamically to implement a rotating priority.

In addition, grouping of YCC entry locations can occur in one of the following two ways:

- Group. In the group scheme, all SCCs are grouped together at the top of the priority table, ahead of most other CPM interrupt sources. This scheme is ideal for applications where all SCCs, FCCs, and MCCs function at a very high data rate and interrupt latency is very important.
- Spread. In the spread scheme, priorities are spread over the table so other sources can have lower interrupt latencies. This scheme is also programmed in the SICR but cannot be changed dynamically.

21.2.2 Highest Priority Interrupt

In addition to the FCC/MCC/SCC relative priority option, SICR[HP] can be used to specify one interrupt source as having highest priority. This interrupt remains within the same interrupt level as the other interrupt controller interrupts, but is serviced before any other interrupt in the table.

If the highest priority feature is not used, select the interrupt request with the highest priority. SICR[HP] can be updated dynamically to allow the user to change a normally low priority source into a high priority-source for a certain period.

21.3 Masking Interrupt Sources

By programming the CPM interrupt mask registers, SIMR_H and SIMR_L, the user can mask interrupt requests to the core. Each SIMR bit corresponds to an interrupt source. To enable an interrupt, set the corresponding SIMR bit. When a masked interrupt source has a pending interrupt request, the corresponding SIPNR bit is set, even though the interrupt is not generated to the core. The user can mask all interrupt sources to implement a polling interrupt servicing scheme.

When an interrupt source has multiple interrupting events, the user can individually mask these events by programming a mask register within that block. Table 21-1 shows which

interrupt sources have multiple interrupting events. Figure 21-2 shows an example of how the masking occurs, using an SCC as an example.

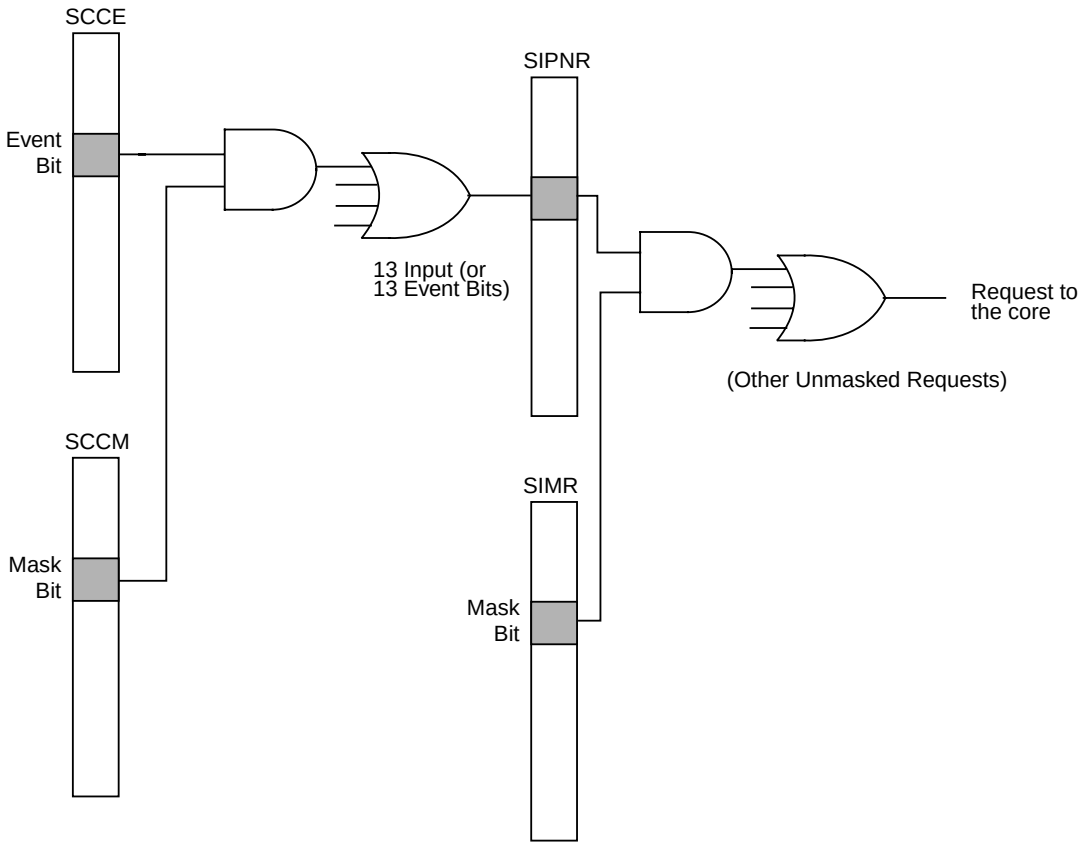


Figure 21-2. Interrupt Request Masking

21.4 CPM Interrupt Vector Generation and Calculation

Pending unmasked interrupts are presented to the core in order of priority. The interrupt vector that allows the core to locate the interrupt service routine is made available to the core by reading SIVEC. The interrupt controller passes an interrupt vector corresponding to the highest-priority, unmasked, pending interrupt. Table 21-2 lists encodings for the six low-order bits of the interrupt vector.

Table 21-2. Encoding the Interrupt Vector

Interrupt Number	Interrupt Source Description	Interrupt Vector
0	Error (No interrupt)	0b00_0000
1	I^2C	0b00_0001
2	SPI	0b00_0010

Table 21-2. Encoding the Interrupt Vector (continued)

Interrupt Number	Interrupt Source Description	Interrupt Vector
3	RISC Timers	0b00_0011
4	Reserved	0b00_0100
5	Reserved	0b00_0101
6	Reserved	0b00_0110
7	Reserved	0b00_0111
8	Reserved	0b00_1000
9	Reserved	0b00_1001
10	SDMA	0b00_1010
11	Reserved	0b00_1011
12	Timer1	0b00_1100
13	Timer2	0b00_1101
14	Timer3	0b00_1110
15	Timer4	0b00_1111
16-31	Reserved	0b01_0000–01_1111
32	FCC1	0b10_0000
33	FCC2	0b10_0001
34	FCC3	0b10_0010
35	Reserved	0b10_0011
36	MCC1	0b10_0100
37	MCC2	0b10_0101
38	Reserved	0b10_0110
39	Reserved	0b10_0111
40	SCC1	0b10_1000
41	SCC2	0b10_1001
42	SCC3	0b10_1010
43	SCC4	0b10_1011
44	TC Layer	0b10_1100
45–47	Reserved	0b10_11xx
48	PC15	0b11_0000
49	PC14	0b11_0001
50	PC13	0b11_0010

Table 21-2. Encoding the Interrupt Vector (continued)

Interrupt Number	Interrupt Source Description	Interrupt Vector
51	PC12	0b11_0011
52	PC11	0b11_0100
53	PC10	0b11_0101
54	PC9	0b11_0110
55	PC8	0b11_0111
56	PC7	0b11_1000
57	PC6	0b11_1001
58	PC5	0b11_1010
59	PC4	0b11_1011
60	PC3	0b11_1100
61	PC2	0b11_1101
62	PC1	0b11_1110
63	PC0	0b11_1111

Note that the interrupt vector table differs from the interrupt priority table in only two ways:

- FCC, SCC, and MCC vectors are fixed; they are not affected by the SCC group mode, spread mode, or the relative priority order of the FCCs, SCCs, and MCC.
- An error vector exists as the last entry in Table 21-2. The error vector is issued when no interrupt is requesting service.

21.4.1 Port C External Interrupts

There are 16 external interrupts, coming from the parallel I/O port C pins, PC[0–15]. When one of these pins is configured as an input, a change according to the CPM external interrupt control register (SIEXR) causes an interrupt request signal to be sent to the interrupt controller. PC[0–15] lines can be programmed to assert an interrupt request upon any change. Each port C line asserts a unique interrupt request to the interrupt pending register and has a different internal interrupt priority level within the interrupt controller.

Requests can be masked independently in the interrupt mask register (SIMR). Notice that the global SIMR is cleared on system reset so pins left floating do not cause false interrupts.

21.5 CPM Interrupt Programming Model

Interrupt controller registers. These registers control configuration, prioritization, and masking of interrupts. They also include registers for determining the interrupt sources. These registers are described in Section 21.5.1, “Interrupt Controller Registers.”

21.5.1 Interrupt Controller Registers

There are seven interrupt controller registers, described in the following sections:

- Section 21.5.1.1, “CPM Interrupt Configuration Register (SICR)”
- Section 21.5.1.2, “CPM Interrupt Priority Registers (SCPRR_H and SCPRR_L)”
- Section 21.5.1.3, “CPM Interrupt Pending Registers (SIPNR_H and SIPNR_L)”
- Section 21.5.1.4, “CPM Interrupt Mask Registers (SIMR_H and SIMR_L)”
- Section 21.5.1.5, “CPM Interrupt Vector Register (SIVEC)”
- Section 21.5.1.6, “CPM External Interrupt Control Register (SIEXR)”

21.5.1.1 CPM Interrupt Configuration Register (SICR)

The CPM interrupt configuration register (SICR), shown in Figure 21-3, defines the highest priority interrupt and whether interrupts are grouped or spread in the priority table, Table 21-1.

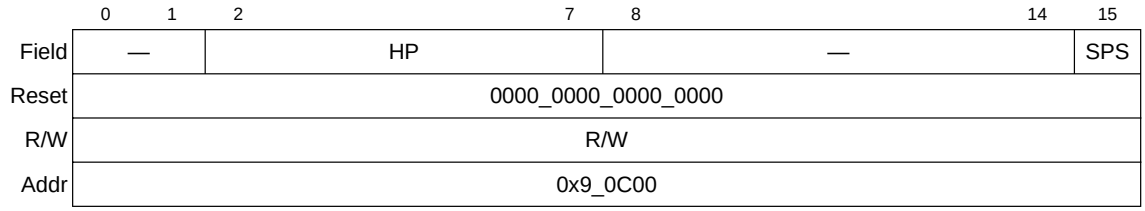


Figure 21-3. CPM Interrupt Configuration Register (SICR)

The SICR register bits are described in Table 21-3.

Table 21-3. SICR Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2–7	HP	Highest priority. Specifies the 6-bit interrupt number of the single interrupt controller interrupt source that is advanced to the highest priority in the table. HP can be modified dynamically. To retain the original priority, program HP to the interrupt number assigned to the interrupt request with the highest priority.
8–14	—	Reserved, should be cleared.
15	SPS	Spread priority scheme. Selects the relative YCC priority scheme. It cannot be changed dynamically. 0 Grouped. The YCCs are grouped by priority at the top of the table. 1 Spread. The YCCs are spread by priority in the table.

21.5.1.2 CPM Interrupt Priority Registers (SCPRR_H and SCPRR_L)

The CPM high interrupt priority register (SCPRR_H), shown in Figure 21-4, define priorities between the FCCs and MCCs.

	0–2	3–5	6–8	9–11	12	13–15
Field	XC1P	XC2P	XC3P	XC4P	—	—
Reset	000	001	010	011	0	000
R/W	R/W					
Addr	0x9_0C14					

	16–18	19–21	22–24	25–27	28	29–31
Field	XC5P	XC6P	XC7P	XC8P	—	—
Reset	100	101	110	111	0	000
R/W	R/W					
Addr	0x9_0C16					

Figure 21-4. CPM High Interrupt Priority Register (SCPRR_H)

Table 21-4 describes SCPRR_H fields.

Table 21-4. SCPRR_H Field Descriptions

Bits	Name	Description
0–2	XC1P–XCC1	Priority order. Defines which FCC/MCC asserts its request in the XCC1 priority position. The user should not program the same FCC/MCC to more than one priority position (1–8). These bits can be changed dynamically. 000 FCC1 asserts its request in the XCC1 position. 001 FCC2 asserts its request in the XCC1 position. 010 FCC3 asserts its request in the XCC1 position. 011 XCC1 position not active. 100 MCC1 asserts its request in the XCC1 position. 101 MCC2 asserts its request in the XCC1 position. 110 XCC1 position not active. 111 XCC1 position not active.
3–12	XC2P–XC8P	Same as XC1P, but for XCC2–XCC8
13–15	—	Reserved, should be cleared.

The CPM low interrupt priority register (SCPRR_L), shown in Figure 21-5, defines prioritization of SCCs and the ATM TC layer.

	0–2	3–5	6–8	9–11	12–15
Field	YC1P	YC2P	YC3P	YC4P	—
Reset	000	001	010	011	0000
R/W	R/W				
Addr	0x9_0C18				

	16–18	19–21	22–24	25–27	28–31
Field	YC5P	YC6P	YC7P	YC8P	—
Reset	100	101	110	111	0000
R/W	R/W				
Addr	0x9_0C20				

Figure 21-5. CPM Low Interrupt Priority Register (SCPRR_L)

Table 21-5 describes SCPRR_L fields.

Table 21-5. SCPRR_L Field Descriptions

Bits	Name	Description
0–2	YC1P–YCC1	Priority order. Defines which SCC asserts its request in the YCC1 priority position. Do not program the same SCC to multiple priority positions. This field can be changed dynamically. 000 SCC1 asserts its request in the YCC1 position. 001 SCC2 asserts its request in the YCC1 position. 010 SCC3 asserts its request in the YCC1 position. 011 SCC4 asserts its request in the YCC1 position. 100 TC layer asserts its request in the YCC1 position. 1xx YCC1 position is not active.
3–11	YC2P–YC8P	Same as YC1P, but for YCC2–YCC8
12–15	—	Reserved, should be cleared.

21.5.1.3 CPM Interrupt Pending Registers (SIPNR_H and SIPNR_L)

Each bit in the CPM interrupt pending registers (SIPNR_H and SIPNR_L), shown in Figure 21-6 and Figure 21-7, corresponds to an interrupt source. When an interrupt is received, the interrupt controller sets the corresponding SIPNR bit.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15
Reset	Undefined (The User Should Write 1s To Clear These Bits Before using)															
R/W	R/W															
Addr	0x9_0C08															

	16	31
Field	—	
Reset	Undefined (The User Should Write 1s To Clear These Bits Before Using)	
R/W	R/W	
Addr	0x9_0C10	

Figure 21-6. SIPNR_H Fields

Figure 21-7 shows SIPNR_L fields.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	15
Field	FCC1	FCC2	FCC3	—	MCC1	MCC2	—		SCC1	SCC2	SCC3	SCC4	TC	—	
Reset	0000_0000_0000_0000[1]														
R/W	R/W														
Addr	0x9_0C0C														

	16	17	18	19	24	25	26	27	28	29	30	31
Field	I^2C	SPI	RTT	—		SDMA	—	TIMER 1	TIMER 2	TIMER 3	TIMER 4	—
Reset	0000_0000_0000_000[1]											0[1]
R/W	R/W											
Addr	0x9_0C0E											

[1] These fields are zero after reset because their corresponding mask register bits are cleared (disabled).

Figure 21-7. SIPNR_L Fields

When a pending interrupt is handled, the user clears the corresponding SIPNR bit. However, if an event register exists, the unmasked event register bits should be cleared instead, causing the SIPNR bit to be cleared.

SIPNR bits are cleared by writing ones to them. Because the user can only clear bits in this register, writing zeros to this register has no effect.

Note that the SCC/FCC/MCC SIPNR bit positions are not changed according to their relative priority.

21.5.1.4 CPM Interrupt Mask Registers (SIMR_H and SIMR_L)

Each bit in the CPM interrupt mask register (SIMR) corresponds to a interrupt source. The user masks an interrupt by clearing and enables an interrupt by setting the corresponding SIMR bit. When a masked interrupt occurs, the corresponding SIPNR bit is set, regardless of the SIMR bit although no interrupt request is passed to the core.

If an interrupt source requests interrupt service when the user clears its SIMR bit, the request stops. If the user sets the SIMR bit later, a previously pending interrupt request is processed by the core, according to its assigned priority. The SIMR can be read by the user at any time.

Figure 21-8 shows the SIMR_H register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	PC0	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_0C1C															

	16	31
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x9_0C1E	

Figure 21-8. SIMR_H Register

Figure 21-9 shows SIMR_L.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	15
Field	FCC1	FCC2	FCC3	—	MCC1	MCC2	—		SCC1	SCC2	SCC3	SCC4	TC	—	
Reset	0000_0000_0000_0000														
R/W	R/W														
Addr	0x9_0C20														

	16	17	18	19	24	25	26	27	28	29	30	31
Field	I^2C	SPI	RTT	—		SDMA	—	TIMER 1	TIMER 2	TIMER 3	TIMER 4	—
Reset	0000_0000_0000_0000											
R/W	R/W											
Addr	0x9_0C22											

Figure 21-9. SIMR_L Register

Note the following:

- SCC/MCC/FCC SIMR bit positions are not affected by their relative priority.
- The user can clear pending register bits that were set by multiple interrupt events only by clearing all unmasked events in the corresponding event register.
- If an SIMR bit is masked at the same time that the corresponding SIPNR bit causes an interrupt request to the core, the error vector is issued (if no other interrupts pending). Thus, the user should always include an error vector routine, even if it contains only an **rfi** instruction. The error vector cannot be masked.

21.5.1.5 CPM Interrupt Vector Register (SIVEC)

The CPM interrupt vector register (SIVEC), shown in Figure 21-10, contains an 8-bit code representing the unmasked interrupt source of the highest priority level.

	0–5	6	7	8	9	10	11	12	13	14	15
Field	Interrupt Code	0	0	0	0	0	0	0	0	0	0
Reset	0000_0000_0000_0000										
R/W	R										
Addr	0x9_0C04										

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0000_0000_0000_0000															
R/W	R															
Addr	0x9_0C06															

Figure 21-10. CPM Interrupt Vector Register (SIVEC)

The SIVEC can be read as either a byte, half word, or a word. When read as a byte, a branch table can be used in which each entry contains one instruction (branch). When read as a half word, each entry can contain a full routine of up to 256 instructions. The interrupt code is defined such that its two lsbs are zeroes, allowing indexing into the table, as shown in Figure 21-11.

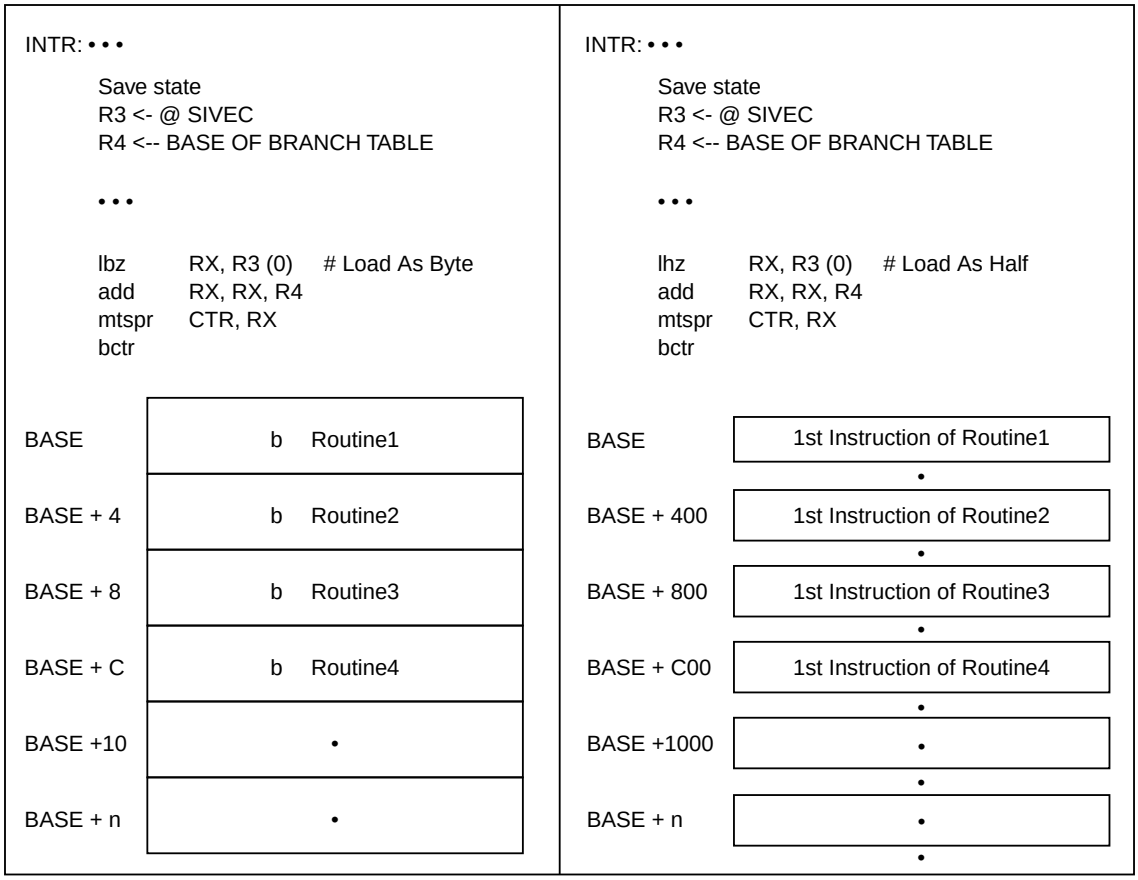


Figure 21-11. Interrupt Table Handling Example

21.5.1.6 CPM External Interrupt Control Register (SIEXR)

Each defined bit in the CPM external interrupt control register (SIEXR), shown in Figure 21-12, determines whether the corresponding port C line asserts an interrupt request upon either a high-to-low change or any change on the pin. External interrupts can come from port C (PC[0-15]).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	EDPC 0	EDPC 1	EDPC 2	EDPC 3	EDPC 4	EDPC 5	EDPC 6	EDPC 7	EDPC 8	EDPC 9	EDPC 10	EDPC 11	EDPC 12	EDPC 13	EDPC 14	EDPC 15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_0C24															

	16	31
Field	—	
Reset	0000_0000_0000_0000	
R/W	R/W	R
Addr	0x9_0C26	

Figure 21-12. CPM External Interrupt Control Register (SIEXR)

Table 21-6 describes SIEXR fields.

Table 21-6. SIEXR Field Descriptions

Bits	Name	Description
0–15	EDPCx	Edge detect mode for port Cx. The corresponding port C line (PCx) asserts an interrupt request according to the following: 0 Any change on PCx generates an interrupt request. 1 High-to-low change on PCx generates an interrupt request.

Chapter 22
Serial Interface with Time-Slot Assigner

Figure 22-1 shows a block diagram of the TSA. Two SI blocks in the MPC8560 (SI1 and SI2), can be programmed to handle eight TDM lines concurrently with the same flexibility described in this manual. TDM channels on SI1 are referred to as TDMa1, TDMb1, TDMc1, TDMd1; TDM channels on SI2 are TDMa2, TDMb2, TDMc2, TDMd2.

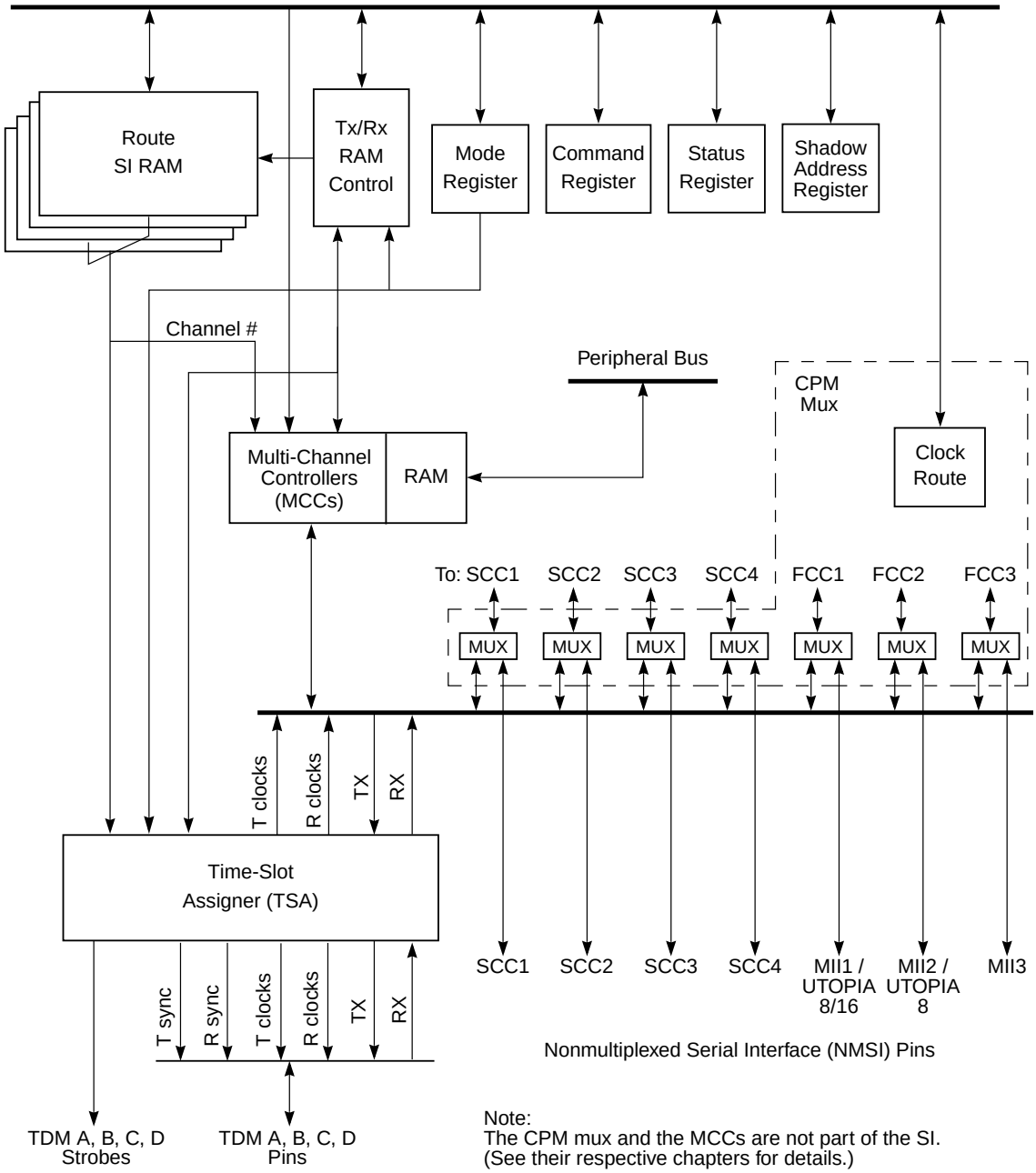


Figure 22-1. SI Block Diagram

If the time-slot assigner (TSA) is not used as intended, it can be used to generate complex wave forms on dedicated output pins. For instance, it can program these pins to implement stepper motor control or variable-duty cycle and period control on-the-fly.

22.1 Features

Each SI has the following features:

- Can connect to four independent TDM channels. Each TDM can be one of the following:
  - T1 or E1 line
  - Integrated services digital network primary rate (PRI)
  - An ISDN basic rate–interchip digital link (IDL) channel in up to four TDM channels—each IDL channel requires support from a separate SCC
  - ISDN basic rate–general circuit interface (GCI). No telecommunications IC (TIC) bus or monitor channel handshake support.
  - E3 or DS3 clear channel on TDMa only (parallel-nibble interface)
  - User-defined interfaces
- Independent, programmable transmit and receive routing paths
- Independent transmit and receive frame syncs allowed
- Independent transmit and receive clocks allowed
- Selection of rising/falling clock edges for the frame sync and data bits
- Supports 1× and 2× input clocks (1 or 2 clocks per data bit)
- Selectable delay (0–3 bits) between frame sync and frame start
- Four programmable strobe outputs and four (2×) clock output pins
- 1- or 8-bit resolution in routing, masking, and strobe selection
- Supports frames up to 16,384 bits long
- Internal routing and strobe selection can be dynamically programmed
- Supports automatic echo and loopback mode for each TDM
- Maximum TDM frequency is determined by the following equations:
  - if DSC = 0 $\frac{\text{CPM Clock}}{2.5}$
  - if DSC = 1 $\frac{\text{CPM Clock}}{2}$

For the MCC route, the SI performs the following features:

- Up to 128 independent communication channels (64-Kbps per channel)
- Arbitrary mapping of any TDM time slots
- Can connect up to four independent TDM channels. Each TDM channel can support up to 128 channels (all four channels can support up to 128 channels together).
- Independent mapping for receive/transmit

- Individual channel echo or loop mode
- Global echo or loop mode through the SI

22.2 Overview

The TSA implements both internal route selection and time-division multiplexing (TDM) for multiplexed serial channels. The TSA supports the serial bus rate and format for most standard TDM buses, including T1 and E1 highways, pulse-code modulation (PCM) highway, and the ISDN buses in both basic and primary rates. The two popular ISDN basic rate buses (interchip digital link (IDL) and general-circuit interface (GCI), also known as IOM-2) are supported.

Because each SI supports four TDMs, it is possible to simultaneously support a combination of up to eight T1 or E1 lines, and basic rate or primary rate ISDN channels.

The TDMa channel can support E3 or DS-3 rates as a clear channel in a parallel-nibble interface.

TSA programming is independent of the protocol used. The serial controllers can be programmed for any synchronous protocol without affecting TSA programming. The TSA simply routes programmed portions of the received data frame from the TDM pins to the target controller, while the target controller handles the received data in the actual protocol.

In its simplest mode, the TSA identifies the frame using one sync pulse and one clock signal provided externally by the user. This can be enhanced to allow independent routing of the receive and transmit data on the TDM. Additionally, the definition of a time-slot need not be limited to 8 bits or even to a single contiguous position within the frame. Finally, the user can provide separate receive and transmit syncs as well as clocks. Figure 22-2 shows example TSA configurations ranging from the simplest to the most complex.

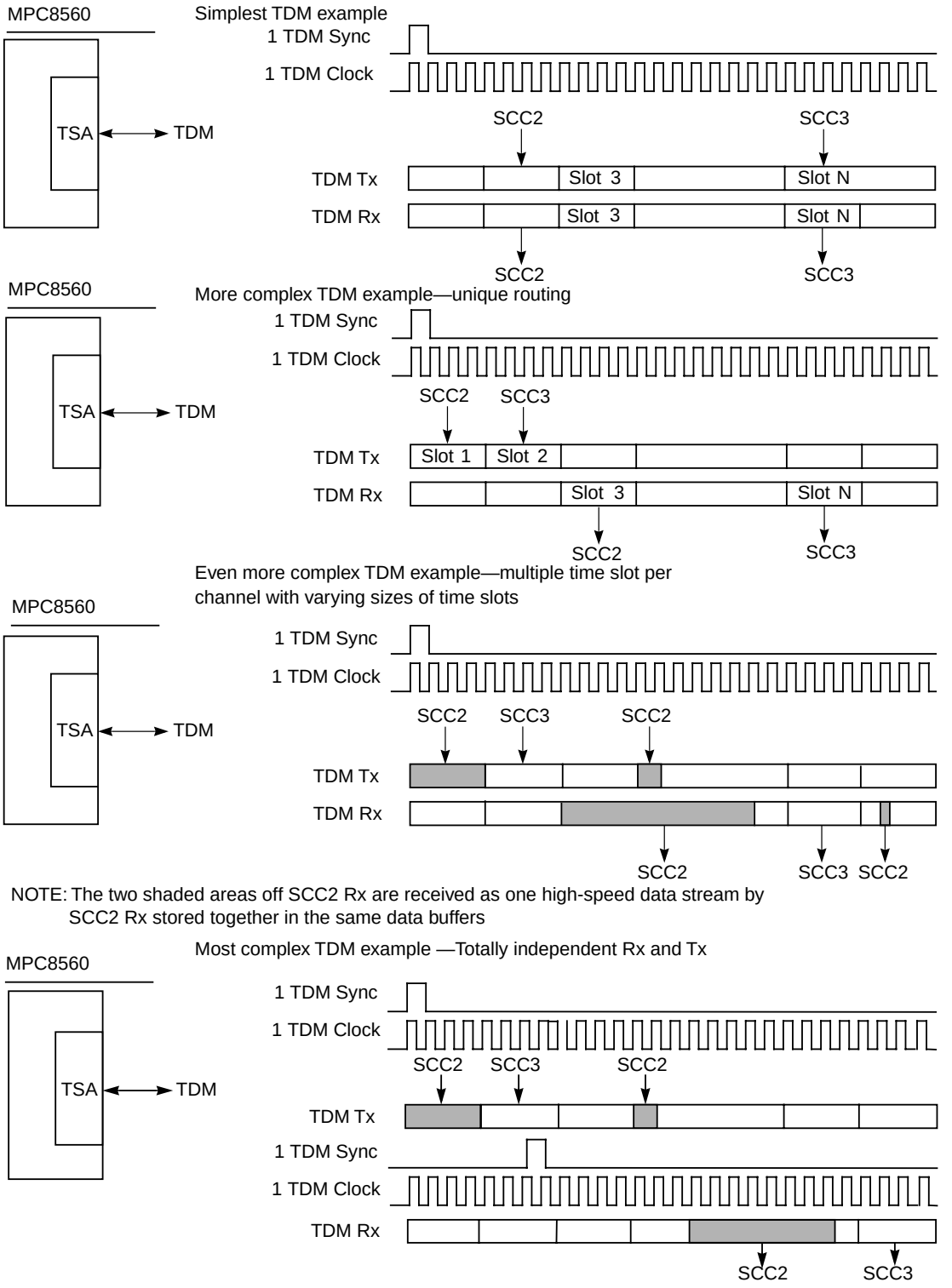
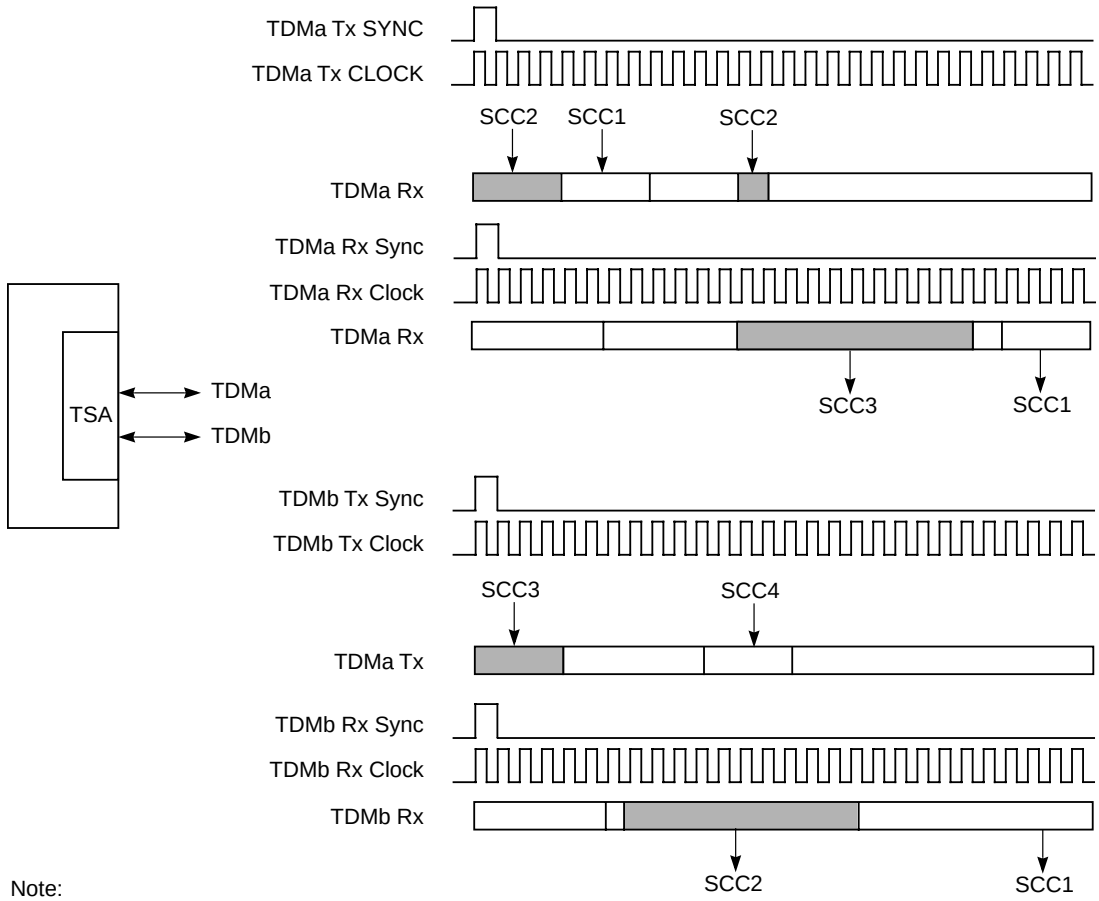


Figure 22-2. Various Configurations of a Single TDM Channel

At its most flexible, the TSA can provide four separate TDM channels, each with independent receive and transmit routing assignments and independent sync pulse and clock inputs. Thus, the TSA can support eight, independent, half-duplex TDM sources, four in reception and four in transmission, using eight sync inputs and eight clock inputs. Figure 22-3 shows a dual-channel example.



Note:
SCCs can receive on one TDM and transmit on another (SCC2 and SCC3).

Figure 22-3. Dual TDM Channel Example

In addition to channel programming, the TSA supports up to four strobe outputs that may be asserted on a bit or byte basis. These strobes are completely independent from the channel routing used by the SCCs. The strobe outputs are useful for interfacing to other devices that do not support the multiplexed interface or for enabling/disabling three-state I/O buffers in a multiple-transmitter architecture. Notice that open-drain programming on the TXDx pins that supports a multiple-transmitter architecture occurs in the parallel I/O block. These strobes can also be used for generating output wave forms to support such applications as stepper-motor control.

Most TSA programming is done in the two 256- × 16-bit SI*x* RAMs. These SI*x* RAMs are directly accessible by the core in the internal register section of the MPC8560 and are not associated with the dual-port RAM. One SI*x* RAM is always used to program the transmit routing; the other is always used to program the receive routing. SI*x* RAMs can be used to define the number of bits/bytes to be routed to the MCC, FCC, or SCC and determine when external strobes are to be asserted and negated.

The size of the SI*x* RAM available for time-slot programming depends on the user's configuration. The user defines how many of the 256 entries are related to each TDM. The resolution of the division is by fractions of 32. If on-the-fly changes are allowed, the SI*x* RAM entries are reduced according to the user's programming. The maximum frame length that can be supported in any configuration is 16,384 bits.

The maximum external serial clock that may be an input to the TSA is CPM CLK/3.

The SI supports two testing modes—echo and loopback.

- The echo mode provides a return signal from the physical interface by retransmitting the signal it has received. The physical interface echo mode differs from the individual FCC or SCC echo mode in that it can operate on the entire TDM signal rather than just on a particular serial channel.
- Loopback mode causes the physical interface to receive the same signal it is sending. The SI loopback mode checks more than the individual serial loopback; it checks both the SI and the internal channel routes.

Note that the flexibility described in the preceding section can be applied to each of the four TDM channels and to all serial interfaces independently.

22.3 Enabling Connections to TSA

Each serial interface can be independently enabled to connect to one of the following: TSA, UTOPIA, MII, or dedicated external pins. Note the following:

- Each FCC can be connected to a dedicated MII or one of four TDMs. FCC1 can also be connected to a 8-/16-bit UTOPIA level-2 interface; FCC2 can also be connected to an 8-bit UTOPIA level-2 interface.
- Each SCC can be connected to one of four TDMs or to its own set of pins.
- The MCC can be connected to one of the four TDMs with different numbers of channels.

The four TDMs are connected to four independent TDM interfaces. Figure 22-4 illustrates the connection between the TSA and the serial interfaces. The connection is made by programming the CPM mux. See Chapter 23, "CPM Multiplexing." Once the connections are made, the exact routing decisions are made in the SI*x* RAM.

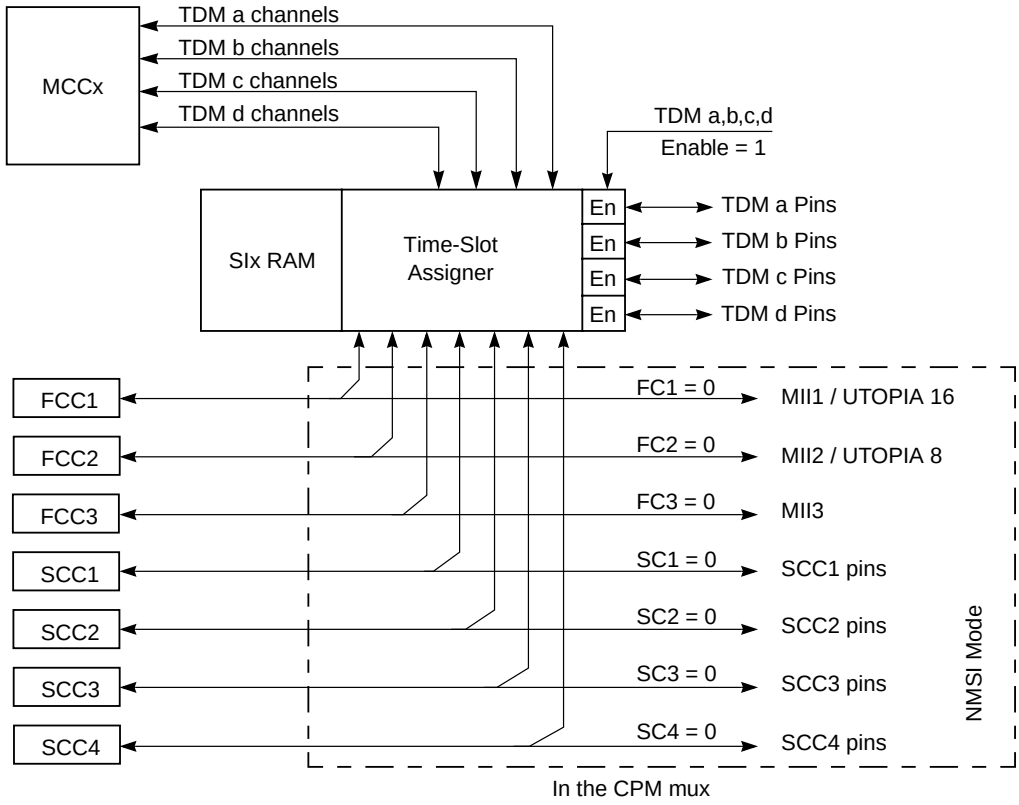


Figure 22-4. Enabling Connections to the TSA

22.4 Serial Interface RAM

Each SI has a transmit RAM and a receive RAM, each with four banks of 64 halfword entries that enable it to control TDM channel routing to all serial devices, including the MCCs. The SI*x* RAMs are uninitialized after power-on reset; unwanted results can occur if the user does not program them before enabling the multiplexed channels.

Each 16-bit SI RAM entry defines the routing of 1–8 bits or bytes at a time. In addition to the routing, up to four strobe pins (logic OR of four strobes in the transmit RAM and four in receive RAM) can be asserted according to the programming of the RAMs. The four SI*x* RAM banks can be configured in many different ways to support various TDM channels. The user can define the size of each SI*x* RAM that is related to a certain TDM channel by programming the starting bank of that TDM. Programming the starting shadow bank address, described in Section 22.5.3, “SIx RAM Shadow Address Registers (SIxRSR),” determines whether this RAM has a shadow for changing SI*x* RAM entries while the TDM channel is active. This reduces the number of available SI*x* RAM entries for that TDM.

22.4.1 One Multiplexed Channel with Static Frames

The example in Figure 22-5 shows one of many possible settings. With this configuration, the SI*x* RAM has 256 entries for transmit data and strobe routing and 256 entries for receive data and strobe routing. This configuration should be chosen only when one TDM is required and the routing on that TDM does not need to be dynamically changed. The number of entries available in the SI*x* RAM is determined by the user.

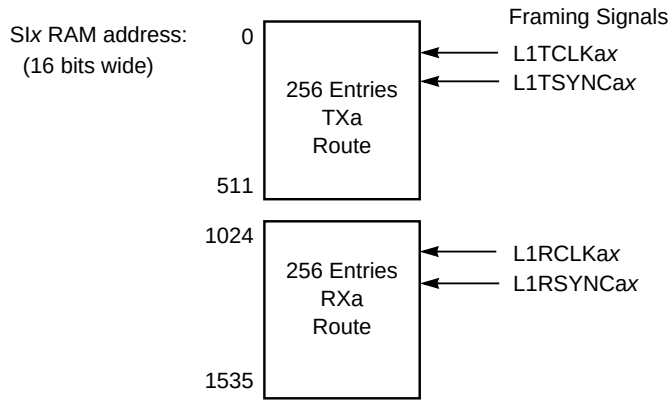


Figure 22-5. One TDM Channel with Static Frames and Independent Rx and Tx Routes

22.4.2 One Multiplexed Channel with Dynamic Frames

In the configuration shown in Figure 22-6, one multiplexed channel has 256 entries for transmit data and strobe routing and 256 entries for receive data and strobe routing. Each RAM has two sections, the current-route RAM and a shadow RAM for changing serial routing dynamically.

After programming the shadow RAM, the user sets SI*x*CMDR[CSR*xn*] for the associated channel. When the next frame sync arrives, the SI automatically exchanges the current-route RAM for the shadow RAM. See Section 22.4.5, “Static and Dynamic Routing.”

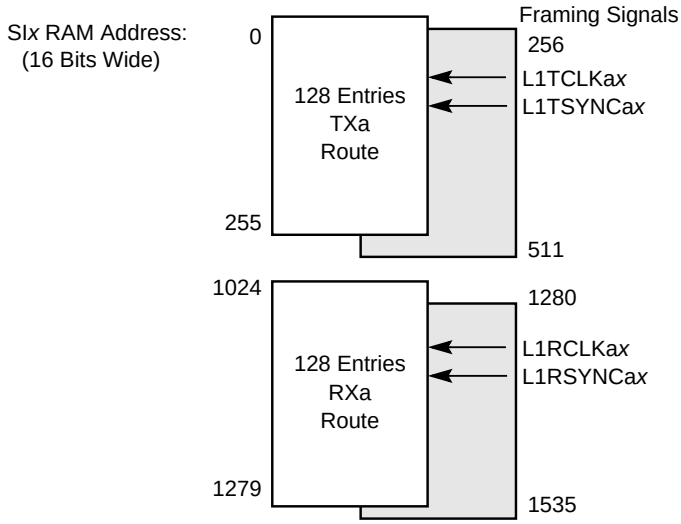


Figure 22-6. One TDM Channel with Shadow RAM for Dynamic Route Change

This configuration should be chosen when only one TDM is needed, but dynamic rerouting may be needed on that TDM. Similarly, for two TDM channels, the number of SI*x* RAM entries are reduced for every TDM channel programmed for shadow mode.

22.4.3 Programming SI*x* RAM Entries

The programming of each entry in the SI*x* RAM determines the routing of the serial bits (or bit groups) and the assertion of strobe outputs. If MCC is set, the entry refers to the corresponding MCC; otherwise, it refers to other serial controllers. Figure 22-7 shows the entry fields for both cases.

	0	1	2	3	4	5	6	7–10	11–13	14	15
Field	MCC = 0	SWTR	SSEL1	SSEL2	SSEL3	SSEL4	—	CSEL	CNT	BYT	LST
	MCC = 1	LOOP/ECHO	SUPER	MCSEL					CNT	BYT	LST
R/W	R/W										
Addr	See Chapter 3, “Memory Map.”										

Figure 22-7. SI*x* RAM Entry Fields

When MCC = 0, the SI*x* RAM entry fields function as described in Table 22-1.

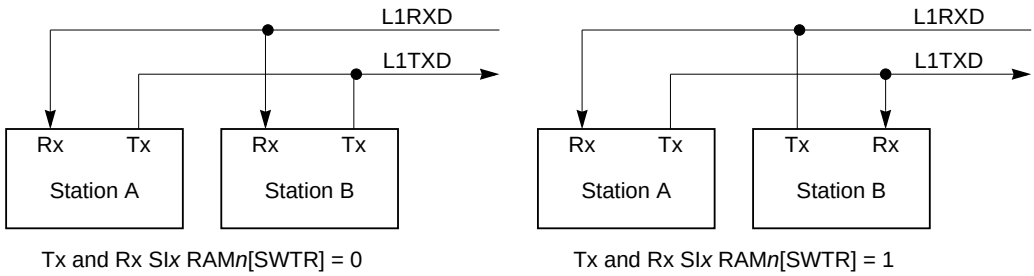
Table 22-1. SI*x* RAM Entry (MCC = 0)

Bits	Name	Description
0	MCC	The entry controls the functionality of the other bits in the SI*x* RAM entry. 0 The entry refers to other serial controllers (FCCs, SCCs, SMC, according to the CSEL field). 1 The entry refers to the MCC.
1	SWTR	Switch Tx and Rx. Valid only in the receive route RAM and ignored in the transmit route RAM. SWTR affects the operation of both L1RXD and L1TXD. SWTR is set only in special situations where the user prefers to receive data from a transmit pin and transmit data on a receive pin. For instance, where devices A and B are connected to the same TDM, each with different time-slots. Normally, there is no opportunity for stations A and B to communicate with each other directly over the TDM, because they both receive the same TDM receive data and transmit on the same TDM transmit signal. 0 Normal operation of L1TXD and L1RXD. 1 Data for this entry is sent on L1RXD and received from L1TXD. See Figure 22-8 for details.
2–5	SSELx	Strobe select. There are four strobes available that can be assigned to the receive RAM and asserted/negated with the received clock of this TDM channel (L1RCLKx). They can also be assigned to the transmit RAM and asserted/negated with the transmit clock of this TDM channel (L1TCLKx). Each bit corresponds to the value the strobe should have during this bit/byte group. There are four strobe pins for all eight strobe bits in the SI*x* RAM entries, so the value on a strobe pin is the logical OR of the Rx and Tx RAM entry strobe bit.s Multiple strobes can be asserted simultaneously. A strobe configured to be asserted in consecutive SI*x* RAM entries remains continuously asserted for both entries. A strobe asserted on the last entry in a table is negated after the last entry is processed. Note: Each strobe is changed with the corresponding RAM clock and is output only if the corresponding parallel I/O is configured as a dedicated pin. If a strobe is programmed to be asserted in more than one set of entries (the SI route entries for more then one TDM channel select the same strobe), the assertion of the strobe corresponds to the logical OR of all possible sources. This use of strobes is not useful for most applications. A given strobe should be selected in only one set of SI*x* RAM entries.
6	—	Reserved, should be cleared.
7–10	CSEL	Channel select. In some MCC cases, when SI frame length is shorter than the gap between sync signals, an underrun condition may appear even though the aggregate serial rate is low. To avoid these cases, add entries with unsupported bits (CSEL = 0000 in SI entry) in the end of the SI frame. Thus SI frame length should match the gap between syncs. 0000The bit/byte group is not supported by the MPC8560. The transmit data pin is three-stated and the receive data pin is ignored. 0001 The bit/byte group is routed to SCC1. 0010 The bit/byte group is routed to SCC2. 0011 The bit/byte group is routed to SCC3. 0100 The bit/byte group is routed to SCC4. 0101 Reserved 0110 Reserved 0111 The bit/byte group is not supported by the MPC8560. 1000 Reserved 1001 The bit/byte group is routed to FCC1. 1010 The bit/byte group is routed to FCC2. 1011 The bit/byte group is routed to FCC3. 11xx Reserved
11–13	CNT	Count. Indicates the number of bits/bytes (according to the BYT bit) that the routing and strobe select of this entry controls. 000 = 1 bit/byte; 111= 8 bits/bytes.

Table 22-1. SI*x* RAM Entry (MCC = 0) (continued)

Bits	Name	Description
14	BYT	Byte resolution 0 Bit resolution. The CNT value indicates the number of bits in this group. 1 Byte resolution. The CNT value indicates the number of bytes in this group.
15	LST	Last entry in the RAM. Whenever SI*x* RAM is used, LST must be set in one of the Tx or Rx entries of each group. Even if all entries of a group are used, this bit must still be set in the last entry. Also note that, to avoid errors in switching to and from shadow SI RAM, the last entry in SI RAM should not be programmed to 1-bit resolution (i.e. CNT = 000 and BYT = 0). 0 Not the last entry in this section of the route RAM. 1 Last entry in this RAM. After this entry, the SI waits for the sync signal to start the next frame. Note that there must be only an even number of entries in an SI*x* RAM frame, because LST is active only in odd-numbered entries (assuming the entry count starts with 0). Therefore, to obtain an even number of entries, an entry may need to be split into two entries.

Figure 22-8 shows how SWTR can be used.



Tx and Rx SI*x* RAM*n*[SWTR] = 0 Tx and Rx SI*x* RAM*n*[SWTR] = 1

Figure 22-8. Using the SWTR Feature

The SWTR option lets station B listen to transmissions from station A and send data to station A. To do this, station B would set SWTR in its receive route RAM. For this entry, receive data is taken from the L1TXD pin and data is sent on the L1RXD pin. If the user wants to listen only to station A transmissions and not send data on L1RXD, the CSEL bits in the corresponding transmit route RAM entry should be cleared to prevent transmission on the L1RXD pin.

Station B can transmit data to station A by setting the SWTR bit of the entry in its receive route RAM. Data is sent on L1RXD rather than L1TXD, according to the transmit route RAM. Note that this configuration could cause collisions with other data on L1RXD unless an available (quiet) time slot is used. To transmit on L1RXD and not receive data on L1TXD, clear the CSEL bits in the receive route RAM.

Note that if the transmit and receive sections of the TDM do not use a common clock source, the SWTR feature can cause erratic behavior. Note also this feature does not work with nibble operation.

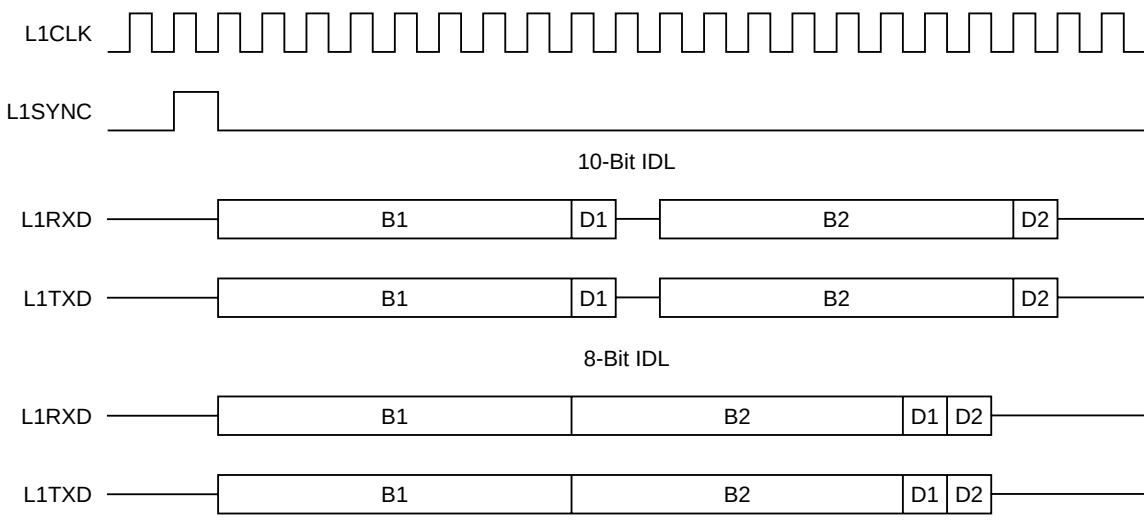
When MCC = 1, the SI*x* RAM entry fields function as described in Table 22-2.

Table 22-2. SI*x* RAM Entry (MCC = 1)

Bits	Name	Description
0	MCC	If MCC =1, the other SI*x* RAM entries in this table are valid.
1	LOOP/ECHO	Channel loopback or echo. 0 Normal mode of operation. 1 Operation depends on the following configurations: In the receive SI*x* RAM, this bit selects loopback mode for this MCC channel. The channel's transmit data is sent to both the receiver's input and to the data output line. In the transmit SI*x* RAM, this bit selects echo mode for this MCC channel. The channel's receive data is sent both to the transmitter's line and to the receiver's input. To use the loop/echo modes, program the receive and transmit SI*x* RAMs identically, except that the LOOP/ECHO bit should be set in only one of the entry pairs; that is, select only one of the modes (echo or loopback, not both) per MCC slot. Also, the receive and transmit clocks must be identical.
2	SUPER	MCC super channel enable. See Section 33.6, "Super-Channel Table." 0 The current entry refers to a regular channel. 1 The current entry refers to a super channel.
3–10	MCSEL	MCC channel select. Indicates the MCC channel the bit/byte group is routed to. 0000_0000 selects channel 0; 1111_1111, selects channel 255. For SI1 use values 0–127 and for SI2 use values 128–255. Note that the channel programming must be coherent with the MCCF; see Section 33.10, "MCC Configuration Registers (MCCFx)."
11–13	CNT	Count. If SUPER = 0 (normal mode), CNT indicates the number of bits/bytes (according to the BYT bit) that the routing select of this entry controls. 000 = 1 bit/byte; 111 = 8 bits/bytes. If SUPER = 1 (MCC super channel), CNT and BYT together indicate whether the current entry is the first byte of the MCC super channel. CNT= 000 and BYT = 1—The current entry is the first byte of this MCC super channel. CNT= 111 and BYT = 0—The current entry is not the first byte of this MCC super channel. Note that because all SI*x* RAM entries relating to super channels must be 1-byte in resolution, only the above two combinations of CNT and BYT are allowed when SUPER = 1.
14	BYT	Byte resolution 0 Bit resolution. The CNT value indicates the number of bits in this group. 1 Byte resolution. The CNT value indicates the number of bytes in this group.
15	LST	Last entry in the RAM. Whenever the SI*x* RAM is used, LST must be set in one of the Tx or Rx entries of each group. Even if all entries of a group are used, LST must still be set in the last entry. Also note that, to avoid errors in switching to and from shadow SI RAM, the last entry in SI RAM should not be programmed to 1-bit resolution (i.e. CNT = 000 and BYT = 0). 0 Not the last entry in this section of the route RAM. 1 Last entry in this RAM. After this entry, the SI waits for the sync signal to start the next frame. Note that there must be only an even number of entries in an SI*x* RAM frame, because LST is active only in odd-numbered entries (assuming the entry count starts with 0). Therefore, to obtain an even number of entries, an entry may need to be split into two entries.

22.4.4 SI*x* RAM Programming Example

This example shows how to program the RAM to support the 10-bit IDL bus. Figure 22-9 shows the 10-bit IDL bus format.



Notes:
1. Clocks are not to scale.
2. L1RQ*x* and L1GR*x* are not shown.

Figure 22-9. IDL Bus Signals

In this example, the TSA supports the B1 channel with SCC2, the D channel with SCC1, the first 4 bits of the B2 channel with an external device (using a strobe to enable the external device), and the last 4 bits of B2 with SCC3. Additionally, the TSA marks the D channel with another strobe signal.

First, divide the frame from the start (the sync) to the end of the frame according to the support that is required:

- 8 bits (B1)—SCC2
- 1 bit (D)—SCC1 + strobe 1
- 1 bit—no support
- 4 bits (B2)—strobe 2
- 4 bits (B2)—SCC3
- 1 bit (D)—SCC1 + strobe 1

Each of these six divisions can be supported by a single SI*x* RAM entry. Thus, six SI*x* RAM entries are needed. See Table 22-3.

Table 22-3. SI*x* RAM Entry Descriptions

Entry Number	SI*x* RAM Entry							
	MCC	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
0	0	0	0000	0010	000	1	0	8-bit SCC2
1	0	0	1000	0001	000	0	0	1-bit SCC1 strobe1

Table 22-3. SI*x* RAM Entry Descriptions (continued)

2	0	0	0000	0000	000	0	0	1-bit no support
3	0	0	0100	0000	011	0	0	4-bit strobe2
4	0	0	0000	0011	011	0	0	4-bit SCC3
5	0	0	1000	0001	000	0	1	1-bit SCC1 strobe1

Note that because IDL requires the same routing for both receive and transmit, an exact duplicate of the above entries should be written to both the receive and transmit sections of the SI*x* RAM. Then SI*x*MR[CRTx] can be used to instruct the SI*x* RAM to use the same clock and sync to simultaneously control both sets of SI*x* RAM entries.

22.4.5 Static and Dynamic Routing

The SI*x* RAM has two operating modes for the TDMs:

- Static routing. The number of SI*x* RAM entries is determined by the banks the user relates to the corresponding TDM and is divided into two parts (Rx and Tx). Three requirements must be met before the new routing takes effect.
  - — All serial devices connected to the TSA must be disabled.
  - — SI routing can be modified.
  - — All appropriate serial devices connected to the TSA must be reenabled.
- Dynamic routing. A TDM's routing definition can be modified while FCCs, MCCs, or SCCs are connected to the TDM. The number of SI*x* RAM entries is determined by the banks the user relates to the corresponding TDM channel and is divided into four parts (Rx, Rx shadow, Tx, and Tx shadow).

Dynamic changes divide portions of the SI*x* RAM into current-route and shadow RAM. Once the current-route RAM is programmed, the TSA and SI channels are enabled, and TSA operation begins. When a change in routing is required, the shadow RAM must be programmed with the new route and SI*x*CMDR[CSR*xn*] must be set. As a result, as soon as the corresponding sync arrives the SI exchanges the shadow RAM with the current-route RAM and resets CSR*xn* to indicate that the operation is complete. At this time, the user may change the routing again. Notice that the original current-route RAM is now the shadow RAM and vice versa. Figure 22-10 shows an example of the shadow RAM exchange process for two TDM channels both with half of the RAM as a shadow.

If for instance one TDM with dynamic changes is programmed to own all four banks, and the shadow is programmed to the last two banks, the initial current-route RAM addresses in the SI*x* RAM are as follows.

- 0–255: TXa route
- 1024–1279: RXa route

The initial shadow RAMs are at addresses:

- 256–511: TXa route
- 1280–1535: RXa route

The user can read any RAM at any time, but for proper SI operation the user must not attempt to write the current-route RAM. The SI*x* status register (SI*x*STR) can be read to find out which part of the RAM is the current-route RAM. The user can also externally connect one of the strobes to an interrupt pin to generate an interrupt on a particular SI*x* RAM entry starting or ending execution by the TSA.

Note that the current-route and shadow SI RAMs of a given TDM*x* should be contiguous; that is, the current-route and shadow SI RAMs of differing TDM*x* should not be interleaved.

An example is shown in Figure 22-10.

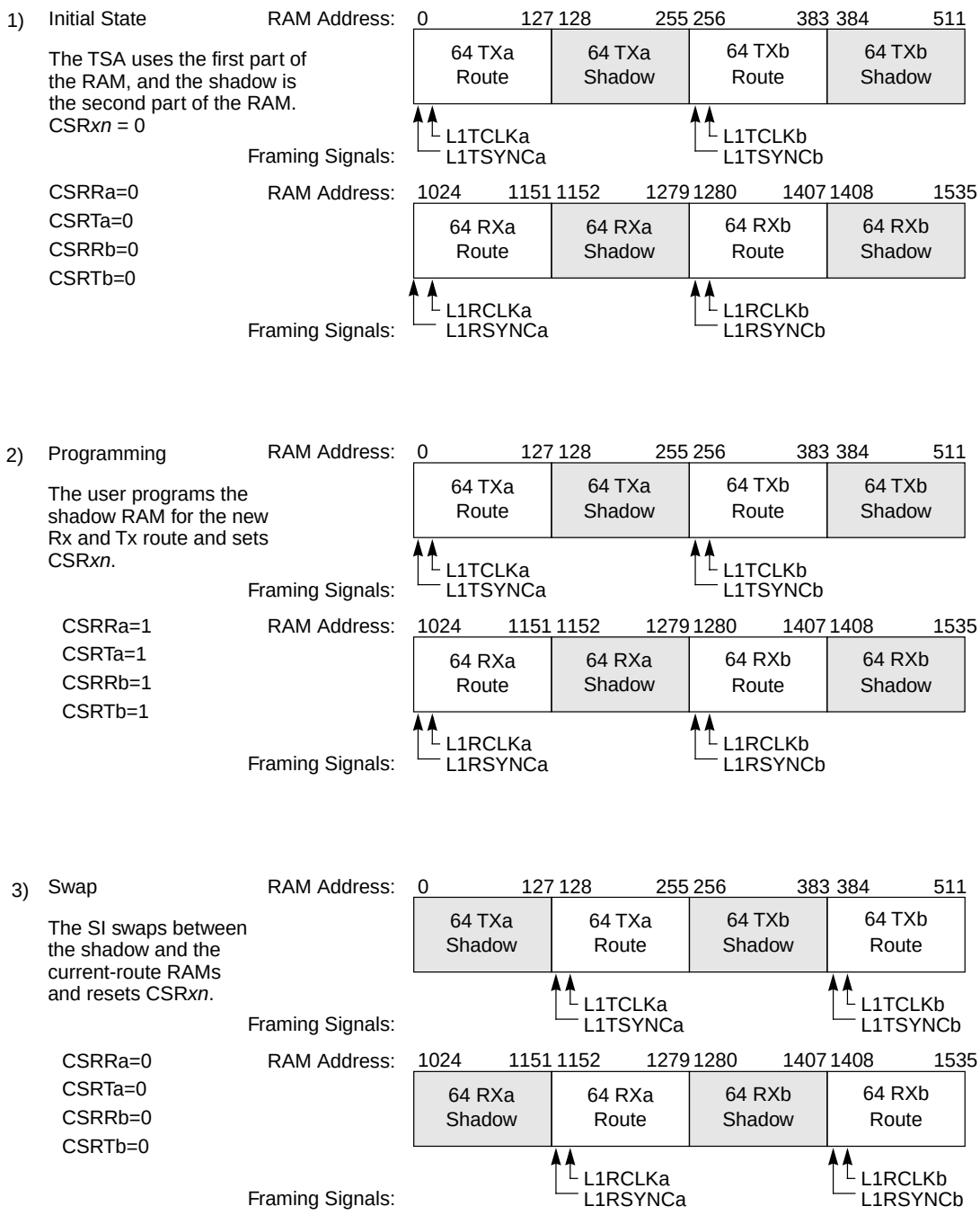


Figure 22-10. Example: SI*x* RAM Dynamic Changes, TDMa and b, Same SI*x* RAM Size

22.5 Serial Interface Registers

The serial interface registers are described in the following sections. The MCC configuration registers, which define the TDM mapping of the MCC channels, are described in Section 33.10, “MCC Configuration Registers (MCCFx).” Note that the programming of SI registers and SI*x* RAM must be coherent with the MCCF programming.

22.5.1 SI Global Mode Registers (SI*x*GMR)

The SI global mode registers (SI*x*GMR), shown in Figure 22-11, defines the activation state of the TDM channels for each SI.

	0	1	2	3	4	5	6	7
Field	STZD	STZC	STZB	STZA	END	ENC	ENB	ENA
Reset	0000_0000							
R/W	R/W							
Addr	0x9_1B28 (SI1GMR), 0x9_1B48 (SI2GMR)							

Figure 22-11. SI Global Mode Registers (SI*x*GMR)

Table 22-4 describes SI*x*GMR.

Table 22-4. SI*x*GMR Field Descriptions

Bit	Name	Description
0–3	STZx	Program L1TXDx to zero for TDM a, b, c or d 0 Normal operation 1 L1TXDx = 0 until serial clocks are available, which is useful for GCI activation. See Section 22.7.1, “Serial Interface GCI Programming.”
4–7	ENx	Enable TDMx. Note that enabling a TDM is the last step in initialization. 0 TDM channel x is disabled. The SI*x* RAMs and routing for TDMx are in a state of reset, but all other SI functions still operate. 1 All TDMx functions are enabled.

22.5.2 SI Mode Registers (SI*x*MR)

There are eight SI mode registers (SI*x*MR), shown in Figure 22-12, one for each TDM channel (SI*x*AMR, SI*x*BMR, SI*x*CMR, and SI*x*DMR). They are used to define SI operation modes and allow the user (with SI*x* RAM) to support any or all of the ISDN channels independently when in IDL or GCI mode. Any extra serial channel can then be used for other purposes.

	0	1–3	4–5	6–7	8	9	10	11	12	13	14–15
Field	—	SADx	SDMx	RFSDx	DSCx	CRTx	SLx	CEx	FEx	GMx	TFSDx
Reset	0000_0000_0000_0000										
R/W	R/W										
Addr	0x9_1B20 (SI1AMR), 0x9_1B22 (SI1BMR), 0x9_1B24 (SI1CMR), 0x9_1B26 (SI1DMR)/ 0x9_1B40 (SI2AMR), 0x9_1B42 (SI2BMR), 0x9_1B44 (SI2CMR), 0x9_1B46 (SI2DMR)										

Figure 22-12. SI Mode Registers (SI*x*MR)

Table 22-5 describes SI*x*MR fields.

Table 22-5. SI*x*MR Field Descriptions

Bits	Name	Description
0	—	Reserved. Should be cleared.
1–3	SADx	Starting bank address for the RAM of TDM a, b, c or d. These three bits define the starting bank address of the SI*x* RAM section that belongs to TDMx channel. Note: As noted previously, the SI*x* RAM contains four banks of 64 entries for receive and four banks of 64 entries for transmit. The starting bank address of each TDM can be programmed with a granularity of 32 entries. The user can put the shadow RAM section of the same TDM on the same bank, but the user cannot put two different TDMs on the same bank. The last entry of a certain TDM is determined by the LST bit in the SI*x* RAM entry. The user must set LST within the entries of SI*x* RAM blocks for every TDM used, that is, before the starting address of the next TDM. 000 first bank, first 32 entries 001 first bank, second 32 entries 010 second bank, first 32entries 011 second bank, second 32 entries 100 third bank, first 32 entries 101 third bank, second 32 entries 110 fourth bank, first 32 entries 111 fourth bank, second 32 entries
4–5	SDMx	SI Diagnostic Mode for TDM a, b, c or d 00 Normal operation 01 Automatic echo. In this mode, the TDM transmitter automatically retransmits the TDM received data on a bit-by-bit basis. The receive section operates normally, but the transmit section can only retransmit received data. In this mode, the L1GRx line is ignored. 10 Internal loopback. In this mode, the TDM transmitter output is internally connected to the TDM receiver input (L1TXDx is connected to L1RXDx). The receiver and transmitter operate normally. The data appears on the L1TXDx pin and in this mode, $\overline{\text{L1RQ}x}$ is asserted normally. The L1GRx line is ignored. 11 Loopback control. In this mode, the TDM transmitter output is internally connected to the TDM receiver input (L1TXDx is connected to L1RXDx). The transmitter output (L1TXDx) and $\overline{\text{L1RQ}x}$ are inactive. This mode is used to accomplish loopback testing of the entire TDM without affecting the external serial lines. Note: In modes 01, 10, and 11, the receive and transmit clocks should be identical.

Table 22-5. SIxMR Field Descriptions (continued)

Bits	Name	Description
6–7	RFSDx	Receive frame sync delay for TDM a, b, c, or d. Determines the number of clock delays between the receive sync and the first bit of the receive frame. Even if CRTx is set, these bits do not control the delay for the transmit frame. 00 No bit delay. The first bit of the frame is transmitted/received on the same clock as the sync; use for GCI. 01 1-bit delay. Use for IDL 10 2-bit delay 11 3-bit delay Figure 22-13 and Figure 22-14 show how these bits are used.
8	DSCx	Double speed clock for TDM a, b, c or d. Some TDMs, such as GCI, define the input clock to be twice as fast as the data rate and this bit controls this option. 0 The channel clock (L1RCLKx and/or L1TCLKx) is equal to the data clock. Use for IDL and most TDM formats. 1 The channel clock rate is twice the data rate. Use for GCI. Note: When an SI is in 2X mode (DSC=1), the SI does not ignore sync signals asserted in the last phase of the last clock cycle of the frame.
9	CRTx	Common receive and transmit pins for TDM a, b, c or d. Useful when the transmit and receive sections of a given TDM use the same clock and sync signals. In this mode, L1TCLKx and L1TSYNCx can be used for their alternate functions. 0 Separate pins. The receive section of this TDM uses L1RCLKx and L1RSYNCx pins for framing and the transmit section uses L1TCLKx and L1TSYNCx for framing. 1 Common pins. The receive and transmit sections of this TDM use L1RCLKx as clock pin of channel x and L1RSYNCx as the receive and transmit sync pin. Use for IDL and GCI. RFSD and TFSD are independent of one another in this mode.
10	SLx	Sync level for TDM a, b, c, or d. 0 The L1RSYNCx and L1TSYNCx signals are active on logic “1”. 1 The L1RSYNCx and L1TSYNCx signals are active on logic “0”.
11	CEx	Clock edge for TDM a, b, c or d. The function depends on DSCx. When DSCx = 0: 0 The data is sent on the rising edge of the clock and received on the falling edge (use for IDL). 1 The data is sent on the falling edge of the clock and received on the rising edge. When DSCx = 1: 0 The data is sent on the rising edge of the clock and received on the rising edge. 1 The data is sent on the falling edge of the clock and received on the falling edge (use for GCI). See Figure 22-15 and Figure 22-16.
12	FEx	Frame sync edge for TDM a, b, c or d. Determines whether L1RSYNCx and L1TSYNCx pulses are sampled with the falling/rising edge of the channel clock. See Figure 22-14, Figure 22-15, Figure 22-16, and Figure 22-17. 0 Falling edge. Use for IDL and GCI. 1 Rising edge.

Table 22-5. SI*x*MR Field Descriptions (continued)

Bits	Name	Description
13	GMx	Grant mode for TDM a, b, c, or d 0 GCI/SCIT mode. 1 IDL mode. A grant mechanism is supported if the corresponding CMXSCR[GR*x*] bit is set. The grant is a sample of L1GRx while L1RSYNCx is asserted. This grant mechanism implies the IDL access controls for transmission on the D channel. See Section 22.6.1, “IDL Interface Programming.”
14–15	TFSDx	Transmit frame sync delay for TDM a, b, c or d. Determines the number of clock delays between the transmit sync and the first bit of the transmit frame. See Figure 22-17. 00 No bit delay. The first bit of the frame is transmitted/received on the same clock as the sync. 01 1-bit delay 10 2-bit delay 11 3-bit delay

Figure 22-13 shows the one-clock delay from sync to data when *x*FSD = 01.

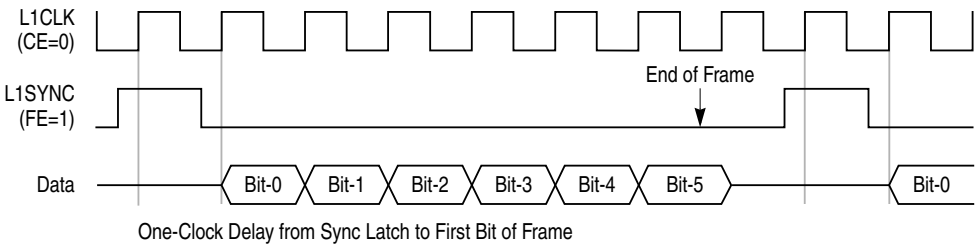


Figure 22-13. One-Clock Delay from Sync to Data (*x*FSD = 01)

Figure 22-14 shows the elimination of the single-clock delay shown in Figure 22-13 by clearing *x*FSD.

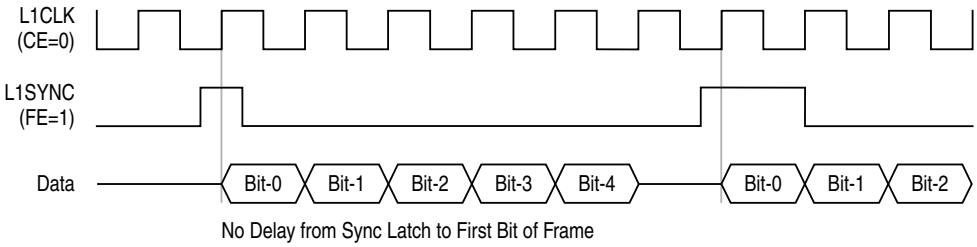


Figure 22-14. No Delay from Sync to Data (*x*FSD = 00)

Figure 22-15 shows the effects of changing FE when CE = 1 with a 1-bit frame sync delay.

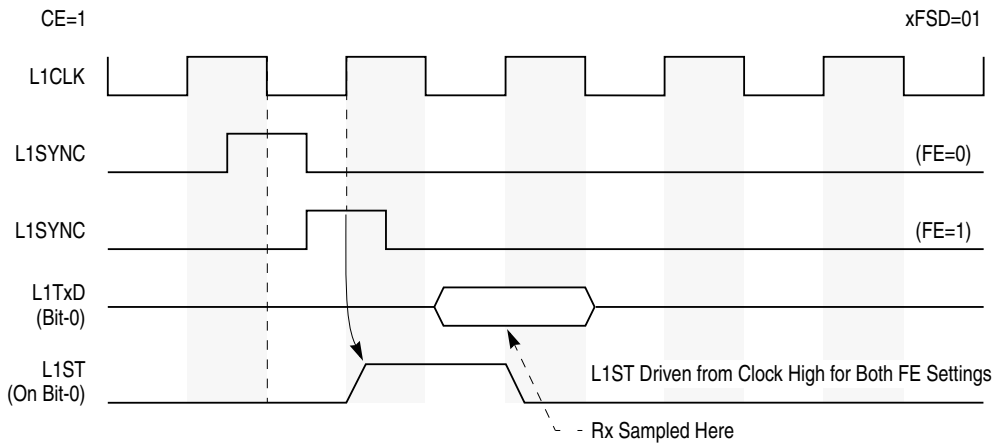


Figure 22-15. Falling Edge (FE) Effect When CE = 1 and *x*FSD = 01

Figure 22-16 shows the effects of changing FE when CE = 0 with a 1-bit frame sync delay.

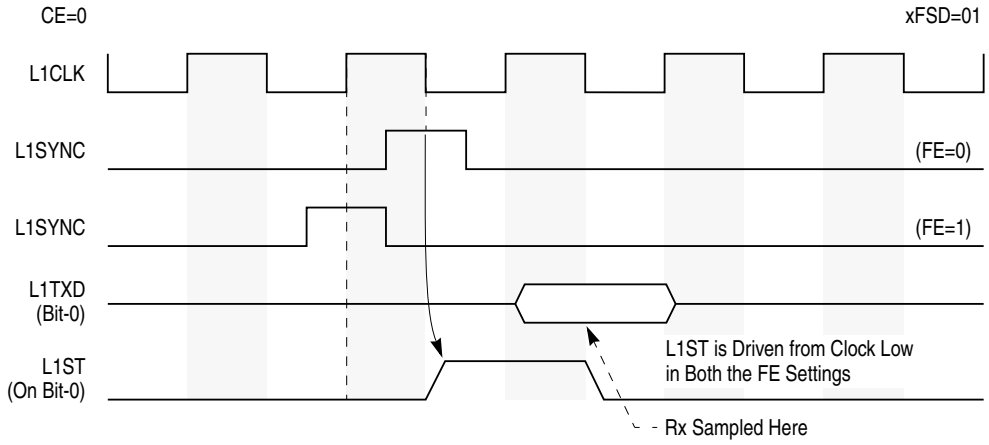


Figure 22-16. Falling Edge (FE) Effect When CE = 0 and *x*FSD = 01

Figure 22-17 shows the effects of changing FE when CE = 1 with no frame sync delay.

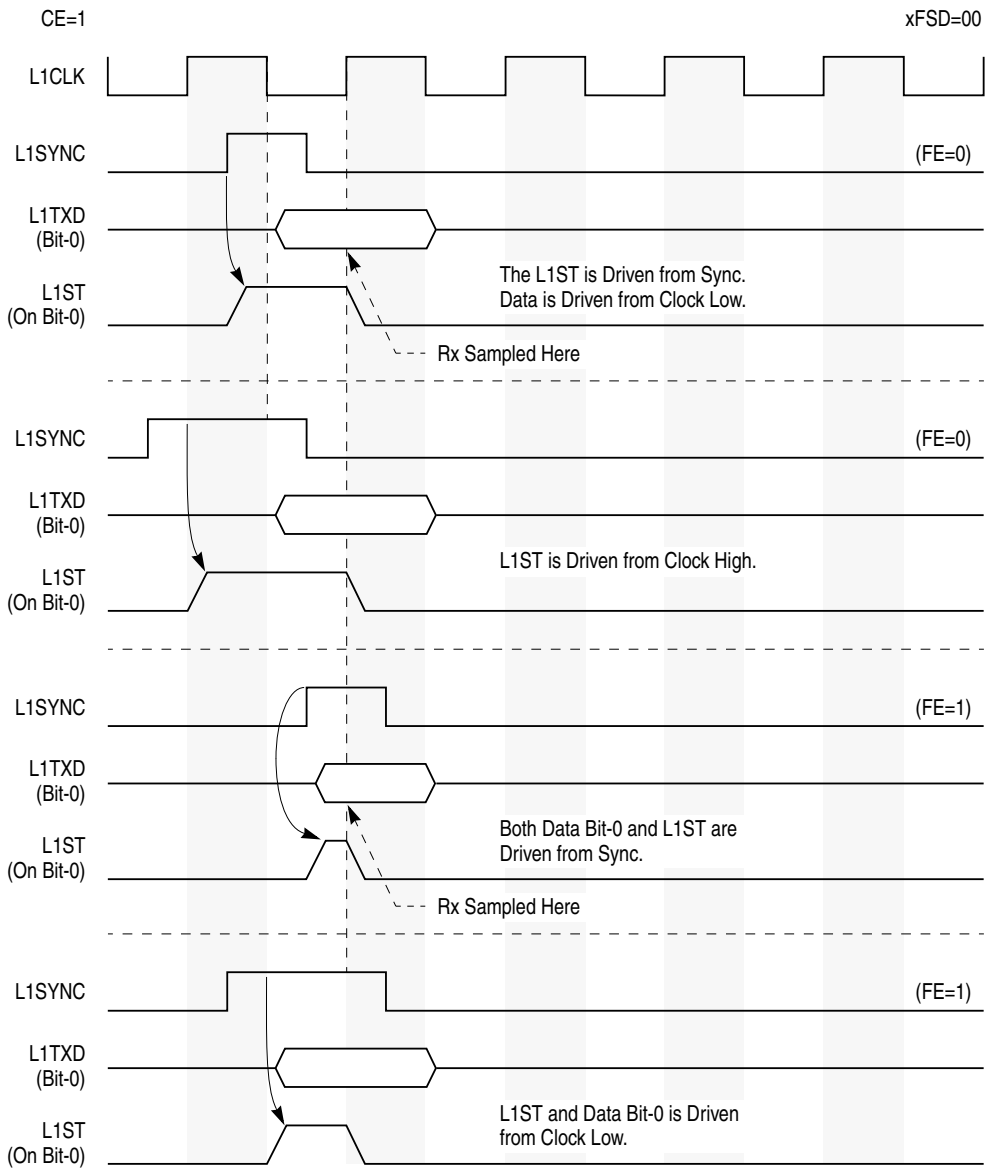


Figure 22-17. Falling Edge (FE) Effect When CE = 1 and *x*FSD = 00

Figure 22-18 shows the effects of changing FE when CE = 0 with no frame sync delay.

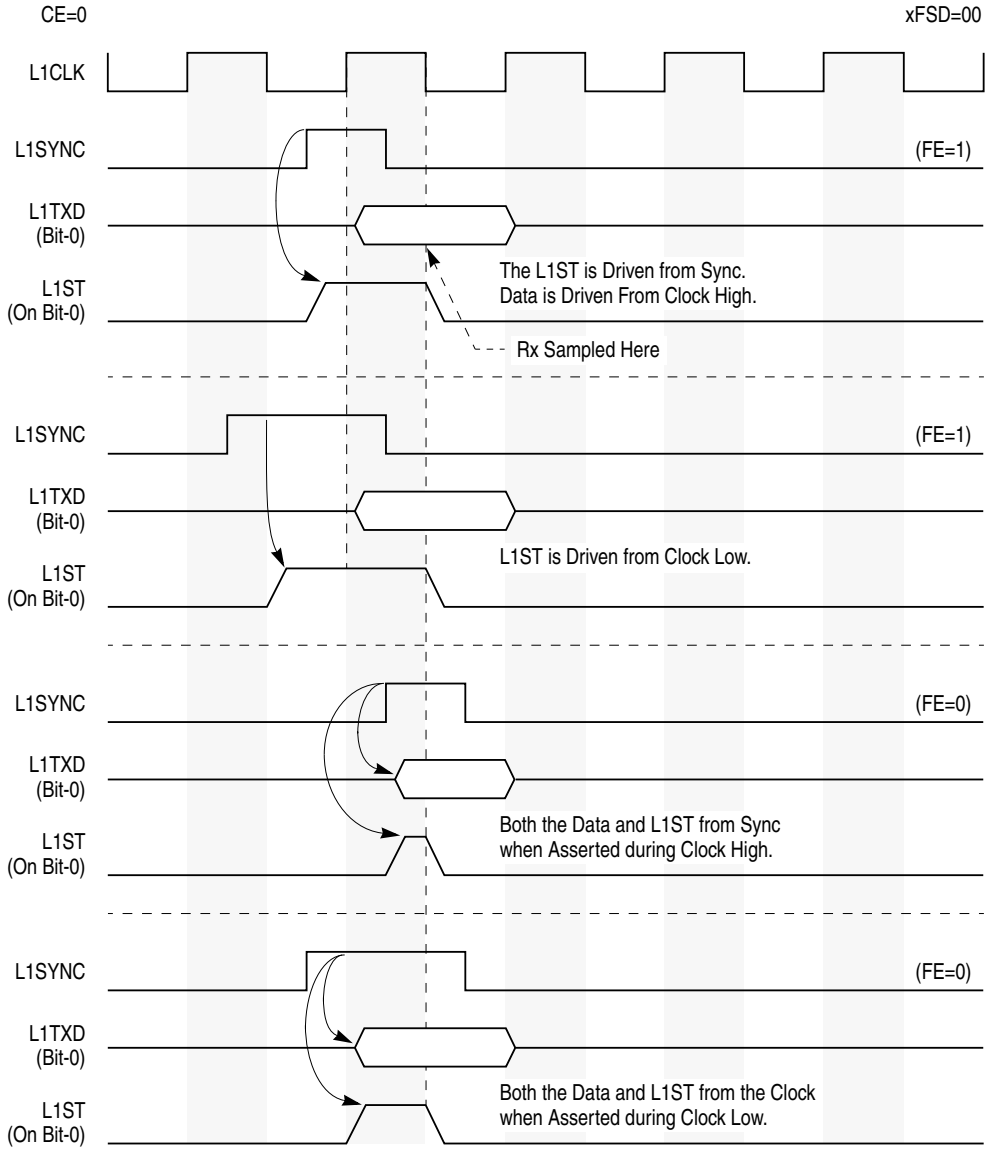
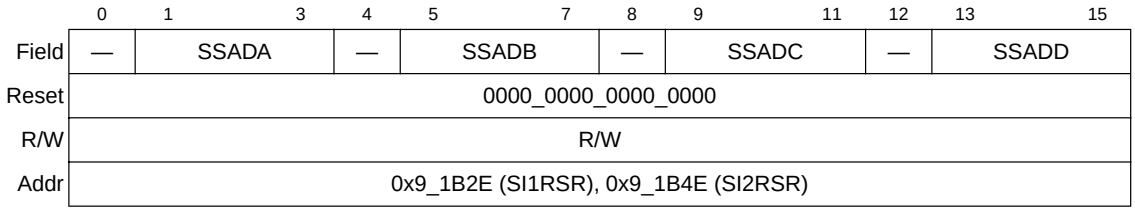


Figure 22-18. Falling Edge (FE) Effect When CE = 0 and *x*FSD = 00

22.5.3 SI*x* RAM Shadow Address Registers (SI*x*RSR)

The SI*x* RAM shadow address registers (SI*x*RSR), shown in Figure 22-19, define the starting addresses of the shadow section in the SI*x* RAM for each of the TDM channels.



	0	1–3	4	5–7	8	9–11	12	13–15
Field	—	SSADA	—	SSADB	—	SSADC	—	SSADD
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x9_1B2E (SI1RSR), 0x9_1B4E (SI2RSR)							

Figure 22-19. SI*x* RAM Shadow Address Registers (SI*x*RSR)

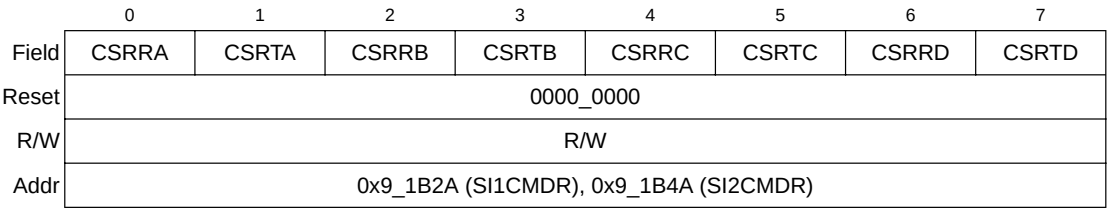
Table 22-6 describes SI*x*RSR fields.

Table 22-6. SI*x*RSR Field Descriptions

Bits	Name	Description
0, 4, 8, 12	—	Reserved. Should be cleared.
1–3, 5–7, 9–11, 13–15	SSAD*x*	Starting bank address for the shadow RAM of TDM a, b, c, or d. Defines the starting bank address of the shadow SI*x* RAM section that belongs to the corresponding TDM channel. Note: As noted before, the SI*x* RAM contain four banks of 64 entries for receive and four banks of 64 entries for transmit. In spite of the above, the starting bank address of each TDM can be programmed by the user in a granularity of 32 entries, but the user cannot put two different TDMs on the same bank. The user can put the shadow RAM section of the same TDM on the same bank. The last entry of a certain TDM frame is determined by the LST bit in the SI*x* RAM entry. The user must set this bit within the entries of SI*x* RAM shadow blocks for every TDM used. That means before the starting address of the next TDM.

22.5.4 SI Command Register (SI*x*CMDR)

The SI command registers (SI*x*CMDR), shown in Figure 22-20, allow the user to dynamically program the SI*x* RAM. When the user sets bits in the SI*x*CMDR, the SI*x* switches to the shadow SI*x* RAM at the end of the current-route RAM programming frame. For more information about dynamic programming, see Section 22.4.5, “Static and Dynamic Routing.”



	0	1	2	3	4	5	6	7
Field	CSRRA	CSRTA	CSRRB	CSRTB	CSRRC	CSRTC	CSRRD	CSRTD
Reset	0000_0000							
R/W	R/W							
Addr	0x9_1B2A (SI1CMDR), 0x9_1B4A (SI2CMDR)							

Figure 22-20. SI Command Register (SI*x*CMDR)

Table 22-7 describes SI*x*CMDR fields.

Table 22-7. SI*x*CMDR Field Description

Bits	Name	Description
0, 2, 4, 6	CSRRx	Change shadow RAM for TDM a, b, c, or d receiver. Set CSRRx causes the SI receiver to replace the current route RAM with the shadow RAM. Set by the user and cleared by the SI. 0 The receiver shadow RAM is not valid. The user can write into the shadow RAM to program a new routing. 1 The receiver shadow RAM is valid. The SI exchanges between the RAMs and take the new receive routing from the receiver shadow RAM. Cleared as soon as the switch has completed.
1, 3, 5, 7	CSRTx	Change shadow RAM for TDM a, b, c, or d transmitter. Set CSRTx causes the SI transmitter to replace the current route RAM with the shadow RAM. Set by the user and cleared by the SI. 0 The transmitter shadow RAM is not valid. The user can write into the shadow RAM to program a new routing. 1 The transmitter shadow RAM is valid. The SI exchanges between the RAMs and take the new transmitter routing from the receiver shadow RAM. Cleared as soon as the switch has completed.

22.5.5 SI Status Registers (SI*x*STR)

The SI status register (SI*x*STR), shown in Figure 22-21, identifies the current-route RAM. SI*x*STR values are valid only when the corresponding SI*x*CMDR bit = 0.

	0	1	2	3	4	5	6	7
Field	CRORA	CROTA	CRORB	CROTB	CRORC	CROTC	CRORD	CROTD
Reset	0000_0000							
R/W	R							
Addr	0x9_1B2C (SI1STR), 0x9_1B4C (SI2STR)							

Figure 22-21. SI Status Registers (SI*x*STR)

Table 22-8 describes SI*x*STR fields.

Table 22-8. SI*x*STR Field Descriptions

Bits	Name	Description
0, 2, 4, 6	CRORx	Current-route original receiver. Determines whether the current-route receiver RAM is the original or the shadow. 0 The current-route receiver RAM is the lower address area. 1 The current-route receiver RAM is the upper address area.
1, 3, 5, 7	CROTx	Current-route original transmitter. Determines whether the current-route transmitter RAM is the original or the shadow. 0 The current-route transmitter RAM is the lower address area. 1 The current-route transmitter RAM is the upper address area.

22.6 Serial Interface IDL Interface Support

The IDL interface is a full-duplex ISDN interface used to connect a physical layer device to the MPC8560. The MPC8560 supports both the basic and primary rate of the IDL bus. In the basic rate of IDL, data on three channels (B1, B2, and D) is transferred in a 20-bit

frame, providing a full-duplex bandwidth of 160 Kbps. The MPC8560 is an IDL slave device that is clocked by the IDL bus master (physical layer device) and has separate receive and transmit sections. Although the MPC8560 has eight TDMs, it can support only four independent IDL buses (limited by the number of serials that support IDL) using separate clocks and sync pulses. Figure 22-22 shows an application with two IDL buses.

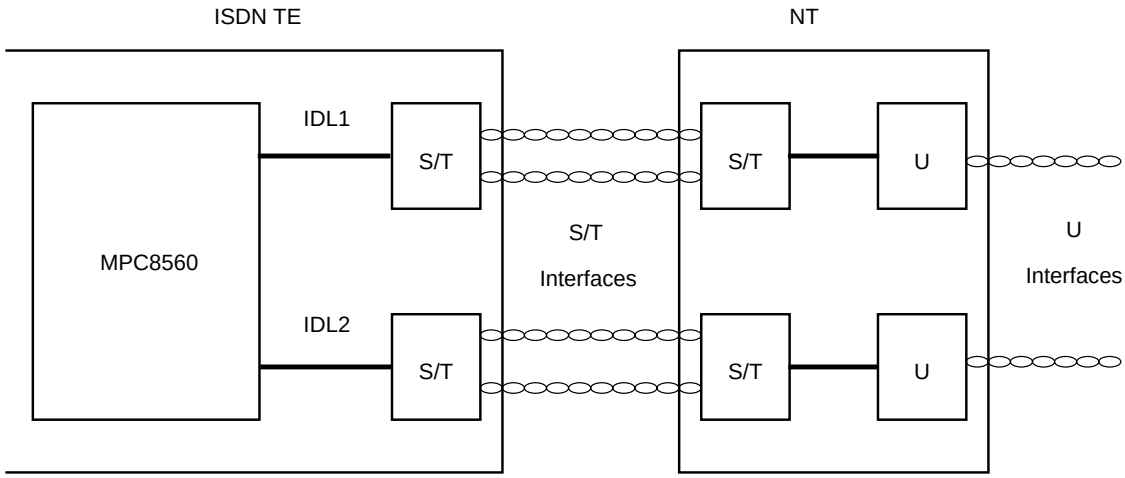


Figure 22-22. Dual IDL Bus Application Example

22.6.1 IDL Interface Programming

To program an IDL interface, first program SI*x*MR[GM*x*] to the IDL grant mode for that channel. If the receive and transmit sections interface to the same IDL bus, set SI*x*MR[CRT*x*] to internally connect the Rx clock and sync signals to the transmit section. Then, program the SI*x* RAM used for the IDL channels to the preferred routing. See Section 22.4.4, “SIx RAM Programming Example.”

Define the IDL frame structure by programming SI*x*MR[*x*FSD*x*] to have a 1-bit delay from frame sync to data, SI*x*MR[FE*x*] to sample on the falling edge, and SI*x*MR[CE*x*] to transmit on the rising edge of the clock. Program the parallel I/O open-drain register so that L1TXD*x* is three-stated when inactive; see Section 44.2.1, “Port Open-Drain Registers (PODRA–PODRD).” To support the D channel, program the appropriate CMXSCR[GR*x*] bit, as described in Section 23.4.5, “CMX SCC Clock Route Register (CMXSCR),” and program the SI*x* RAM entry to route data to the chosen serial controller. The two definitions of IDL, 8 or 10 bits, are implemented by simply modifying the SI*x* RAM programming. In both cases, L1GR*x* is sampled while L1TSYNC*x* is asserted and transferred to the D-channel SCC as a grant indication.

For example, based on the same 10-bit format as in Section 22.4.4, “SIx RAM Programming Example,” implement an IDL bus using SCC1, SCC2, and SCC3 connected to TDMa1 as follows:

1. Program both the Tx and Rx sections of the SI*x* RAM as in Table 22-9.

Table 22-9. SI*x* RAM Entries for an IDL Interface

Entry Number	SI*x* RAM Entry							
	MCC	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
0	0	0	0000	0010	000	1	0	8-bit SCC2
1	0	0	0000	0001	000	0	0	1-bit SCC1
2	0	0	0000	0000	000	0	0	1-bit no support
3	0	0	0000	0011	011	1	0	4-bit SCC3
4	0	0	0000	0011	011	1	0	4-bit SCC3
5	0	0	1000	0001	000	0	1	1-bit SCC1 strobe1

2. CMXSI1CR = 0x00. TDMA receive clock is CLK1.
3. CMXSCR = 0xC040_4000. SCC1, SCC2 and SCC3 are connected to the TSA.
4. SI1AMR = 0x0145. TDMA grant mode is used with 1-bit frame sync delay in Tx and Rx and common receive-transmit mode.
5. Set PPARA[6–9]. Configures L1TXDa[0], L1RXDa[0], L1TSYNCa, and L1RSYNCa.
6. Set PSORA[6–9]. Configures L1TXDa[0], L1RXDa[0], L1TSYNCa, and L1RSYNCa.
7. Set PDIRA[9]. Configures L1TXDa[0].
8. Set PODRA[9]. Configures L1TXDa[0] to an open-drain output.
9. Set PPARC[30,31]. Configures L1TCLKa and L1RCLKa.
10. Clear PDIRC[30,31] Configures L1TCLKa and L1RCLKa.
11. Clear PSORC[30,31]. Configures L1TCLKa and L1RCLKa.
12. Set PPARB[17]. Configures $\overline{\text{L1RQa}}$.
13. Clear PSORB[17]. Configures $\overline{\text{L1RQa}}$.
14. Set PDIRB[17]. Configures $\overline{\text{L1RQa}}$.
15. Set PPARD[13]. Configures L1ST1.
16. Clear PSORD[13]. Configures L1ST1.
17. Set PDIRD[13]. Configures L1ST1.
18. SI1CMDR is not used.
19. SI1STR does not need to be read.

20. Configure the SCC1 for HDLC operation (to handle the LAPD protocol of the D channel), and configure SCC2 and SCC3 as preferred.
21. SI1GMR = 0x01. Enable TDM A (one static TDM).
22. Enable SCC1, SCC2 and SCC3.

22.7 Serial Interface GCI Support

The MPC8560 fully supports the normal mode of the GCI, also known as the ISDN-oriented modular revision 2.2 (IOM-2), and the SCIT. Support for the terminal IC (TIC) bus is not supported.

The GCI bus consists of four lines—two data lines, a clock, and a frame synchronization line. Usually, an 8-kHz frame structure defines the various channels within the 256-kbps data rate. The interface can also be used in a multiplexed frame structure on which up to eight physical layer devices multiplex their GCI channels. In this mode, the data rate would be 2,048 kbps.

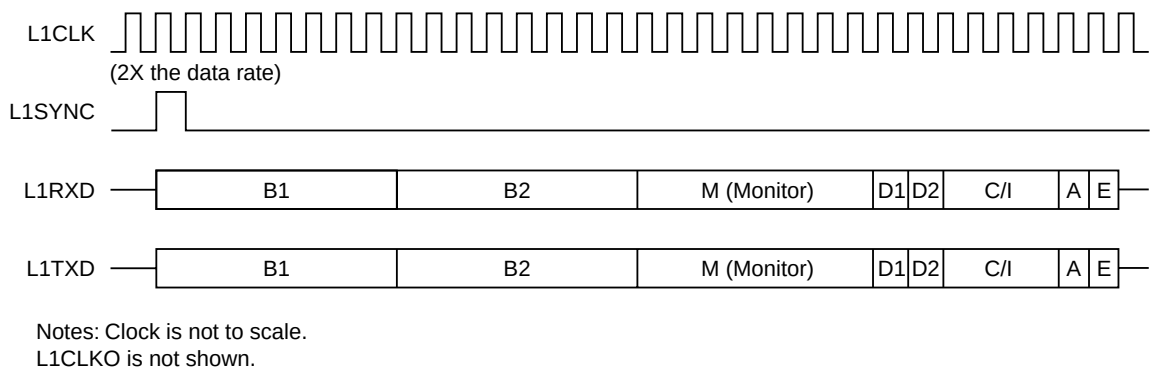
In the GCI bus, the clock rate is twice the data rate. The SI divides the input clock by two to produce the data clock. The MPC8560 also has data strobe lines and the 1× data rate clock L1CLKOx output pins. These signals are used for interfacing devices to GCI that do not support the GCI bus. Table 22-10 describes GCI signals for each transmit and receive channel.

Table 22-10. GCI Signals

Signal	Description
L1RSYNC*x*	Used as a GCI sync signal; input to the MPC8560. This signal indicates that the clock periods following the pulse designate the GCI frame.
L1RCLK*x*	Used as a GCI clock; input to the MPC8560. The L1RCLK*x* signal frequency is twice the data clock.
L1RXD*x*	Used as a GCI receive data; input to the MPC8560.
L1TXD*x*	Used as a GCI transmit data; open-drain output. Valid only for the bits that are supported by the IDL; otherwise, three-stated.
L1CLKO*x*	Optional signal; output from the MPC8560. This 1× clock output is used to clock devices that do not interface directly to the GCI. If the double-speed clock is used, (DSC*x* bit is set in the SI*x*MR), this output is the L1RCLK*x* divided by 2; otherwise, it is simply a 1× output of the L1RCLKx signal.

Note: *x* = a, b, c, and d for TDMa, TDMb, TDMc, and TDMd (for SI1 and SI2).

The GCI bus signals are shown in Figure 22-23.



Notes: Clock is not to scale.
L1CLKO is not shown.

Figure 22-23. GCI Bus Signals

In addition to the 144-Kbps ISDN 2B+D channels, the GCI provides five channels for maintenance and control functions:

- B1 is a 64-Kbps bearer channel
- B2 is a 64-Kbps bearer channel
- M is a 64-Kbps monitor channel
- D is a 16-Kbps signaling channel
- C/I is a 48-Kbps C/I channel (includes A and E bits)

The M channel is used to transfer data between layer 1 devices and the control unit (the CPU); the C/I channel is used to control activation/deactivation procedures or to switch test loops by the control unit.

The GCI supports the CCITT I.460 recommendation as a method for data rate adaptation since it can access each bit of the GCI separately. The current-route RAM specifies which bits are supported by the interface and which serial controller support them. The receiver only receives the bits that are enabled by the SI*x* RAM and the transmitter only transmits the bits that are enabled by the SI*x* RAM and does not drive L1TXDx. Otherwise, L1TXDx is an open-drain output and should be pulled high externally.

22.7.1 Serial Interface GCI Programming

The following sections describe serial interface GCI programming.

22.7.1.1 Normal Mode GCI Programming

The user can program and configure the channels used for the GCI bus interface. First, the SI*x*MR register to the GCI/SCIT mode for that channel must be programmed, using the DSCx, FEx, CEx, and RFSDx bits. This mode defines the sync pulse to GCI sync for framing and data clock as one-half the input clock rate. The user can program more than one channel to interface to the GCI bus. Also, if the receive and transmit section are used

for interfacing the same GCI bus, the user internally connects the receive clock and sync signals to the SI*x* RAM transmit section, using the CRTx bits. The user should then define the GCI frame routing and strobe select using the SI*x* RAM.

When the receive and transmit section uses the same clock and sync signals, these sections should be programmed to the same configuration. Also, the L1TXDx pin in the I/O register should be programmed to be an open-drain output. To support the D channel when there is no possibility of collision, the user should clear the SI*x*MR[GR*x*] bit corresponding to the SCC that supports the D channel.

22.7.1.2 SCIT Programming

For interfacing the GCI/SCIT bus, SI*x*MR must be programmed to the GCI/SCIT mode. The SI*x* RAM is programmed to support a 96-bit frame length and the frame sync is programmed to the GCI sync pulse. For this purpose, the user should program the CRT*x* bits so the receive and transmit sections use the same clock and sync signals

For example, assuming that SCC1 is connected to the D channel, SCC2 to the B1 channel, and SCC3 to the B2 channel, the initialization sequence is as follows:

1. Program both the Tx and Rx sections of the SI*x* RAM as in Table 22-11 beginning at addresses 0 and 1024, respectively.

Table 22-11. SI*x* RAM Entries for a GCI Interface (SCIT Mode)

Entry Number	SI*x* RAM Entry							
	MCC	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
0	0	0	0000	0010	000	1	0	8 Bits SCC2
1	0	0	0000	0011	000	1	0	8 Bits SCC3
2	0	0	0000	0000	000	1	0	Skip 8 Bits
3	0	0	0000	0001	001	0	0	2 Bits SCC1
4	0	0	0000	0000	101	0	0	Skip 6 Bits
5	0	0	0000	0000	110	1	0	Skip 7 bytes
6	0	0	0000	0000	010	0	1	Skip 3 bits

2. SI1AMR = 0x00C0. TDMa is used in double speed clock and common Rx/Tx modes. SCIT mode is used in this example.

   Note: If SCIT mode is not used, delete the last three entries of the SI*x* RAM, divide one entry into two and set the LST bit in the new last entry.

3. CMXSCR = 0xC040_4000. SCC1, SCC2 and SCC3 are connected to the TSA.
4. CMXSI1CR = 0x00. TDMa uses CLK1.

5. Set PPARA[6–9]. Configures L1TXDa[0], L1RXDa[0], L1TSYNCa and L1RSYNCa.
6. Set PSORA[6–9]. Configures L1TXDa[0], L1RXDa[0], L1TSYNCa and L1RSYNCa.
7. Set PDIRA[9]. Configures L1TXDa[0].
8. Set PODRA[9]. Configures L1TXDa[0] to an open-drain output.
9. Set PPARC[30,31]. Configures L1TCLKa and L1RCLKa.
10. Clear PDIRC[30,31]. Configures L1TCLKa and L1RCLKa.
11. Clear PSORC[30,31]. Configures L1TCLKa and L1RCLKa.
12. Set PPARB[17]. Configures L1CLKO and $\overline{\text{L1RQa}}$.
13. Clear PSORB[17]. Configures L1CLKO and $\overline{\text{L1RQa}}$.
14. Set PDIRB[17]. Configures L1CLKO and $\overline{\text{L1RQa}}$.
15. If the 1x GCI data clock is required, set PBPAR bit 16 and PBDIR bit 16 and clear PSORB 16, which configures L1CLKOa as an output.
16. Configure SCC1 for HDLC operation (to handle the LAPD protocol of the D channel). Configure SCC2 and SCC3 as preferred.
17. SI1GMR = 0x11. Enable TDMa (one static TDM), STZ for TDMa.
18. SI1CMDR is not used.
19. SI1STR does not need to be read.
20. Enable the SCC1, SCC2, and SCC3.

Chapter 23
CPM Multiplexing

The CPM multiplexing logic (CMX) connects the physical layer—UTOPIA, MII, modem lines, TDM lines and proprietary serial lines to the FCCs and SCCs. The CMX features the following two modes:

- In NMSI mode, the CMX allows all serial devices to be connected to their own set of individual pins. Each serial device that connects to the external world in this way is said to connect to a nonmultiplexed serial interface (NMSI). In the NMSI configuration, the CMX provides a flexible clocking assignment for each FCC and SCC from a bank of external clock pins and/or internal BRGs.
- In TDM mode, the CMX performs the connection of the serial devices to the SIs for using the time-slot assigner (TSA). This allows any combination of MCCs, FCCs, and SCCs to multiplex data on any of the eight TDM channels. The CMX connects the serial device only to the TSA in the SI*x*. The actual multiplexing of the TDM is made by programming the SI*x* RAM. In TDM mode, all other pins used in NMSI mode are available for other purposes. See Chapter 22, “Serial Interface with Time-Slot Assigner.”

The CMX also allows the user to route the multiple-PHY address to FCC1 or to FCC2 in various combinations, allowing the use of both FCCs in multiple-PHY mode.

NOTE

The CMX serves both SI1 and SI2. When the user programs the CMX to connect a serial device to the SI, the CMX connects that serial device to both SIs. Programming both SIs to use one serial device in the same time slot causes erratic behavior.

Figure 23-1 shows a block diagram of the CMX.

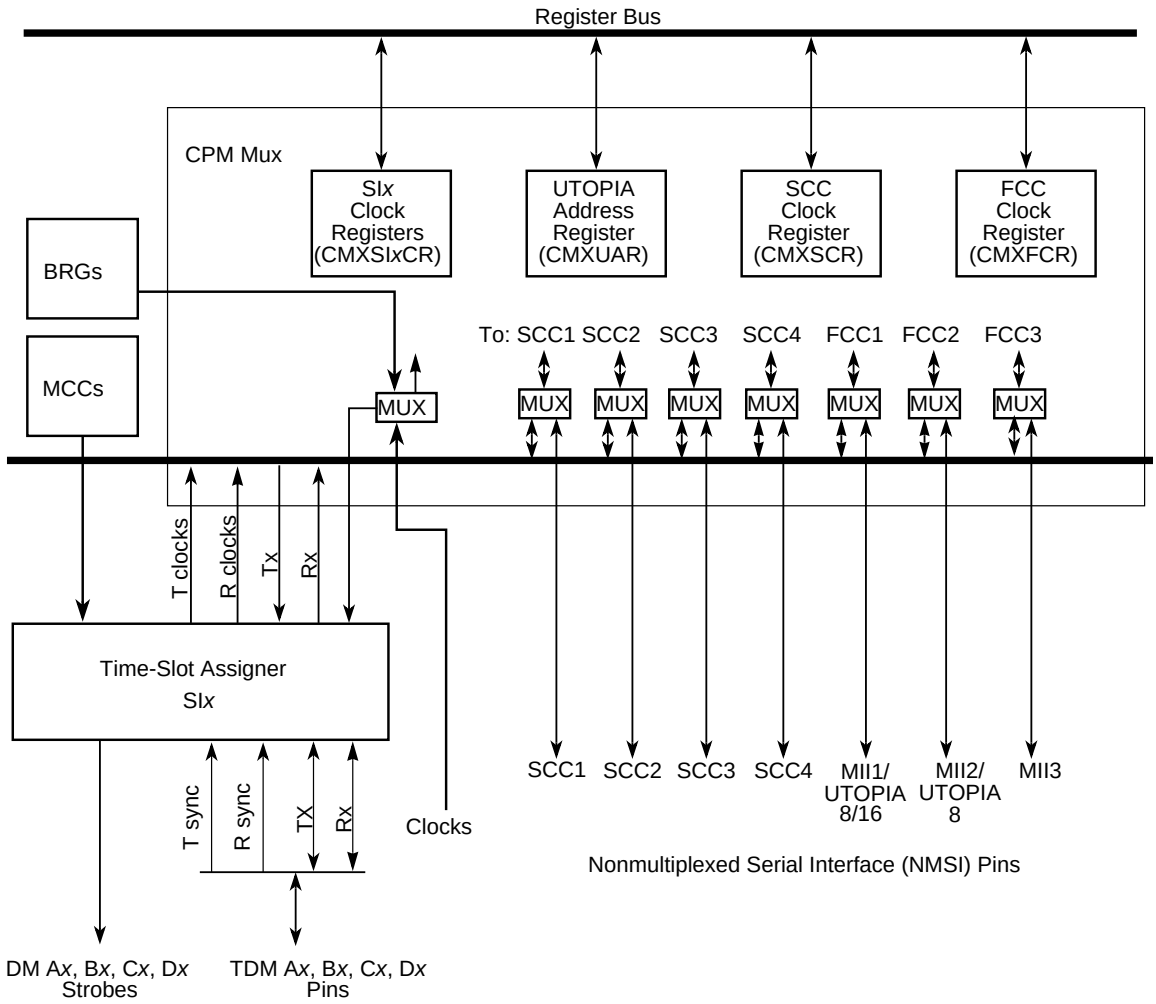


Figure 23-1. CPM Multiplexing Logic (CMX) Block Diagram

23.1 Features

The NMSI mode supports the following:

- Each FCC and SCC can be programmed independently to work with a serial device's own set of pins in a non-multiplexed manner.
- Each FCC can be connected to its own MII (media-independent interface).
- FCC1 can also be connected to an 8- or 16-bit ATM UTOPIA level-2 interface.
- FCC2 can also be connected also to an 8-bit ATM UTOPIA level-2 interface.
- Each SCC can have its own set of modem control pins.
- Each FCC and SCC can be driven from a bank of 20 clock pins or a bank of 8 BRGs.

The multiple-PHY addressing selection supports the following options for FCC1 and FCC2:

- In master mode:
  - FCC1 connect up to 31 PHYs and FCC2 connect up to 7 PHYs
  - FCC1 connect up to 15 PHYs and FCC2 connect up to 15 PHYs
  - FCC1 connect up to 7 PHYs and FCC2 connect up to 31 PHYs
- In slave mode:
  - FCC1 connect up to 31 PHYs and FCC2 connect to 0 PHY
  - FCC1 connect up to 15 PHYs and FCC2 connect up to 1 PHY
  - FCC1 connect up to 7 PHYs and FCC2 connect up to 3 PHYs
  - FCC1 connect up to 3 PHYs and FCC2 connect up to 7 PHYs
  - FCC1 connect up to 1 PHY and FCC2 connect up to 15 PHYs
  - FCC1 connect to 0 PHY and FCC2 connect up to 31 PHYs

23.2 Enabling Connections to TSA or NMSI

Each serial device can be independently enabled to connect to the TSA or to dedicated external pins, as shown in Figure 23-2. Each FCC can be connected to a dedicated MII or to the eight TDMs. FCC1 can also be connected to an 8- or 16-bit UTOPIA level-2 interface. FCC2 can also be connected to an 8-bit UTOPIA level-2 interface. Each SCC can be connected to the eight TDMs or to its own set of pins. Once connections are made to the TSA, the exact routing decisions are made in the SI*x* RAMs.

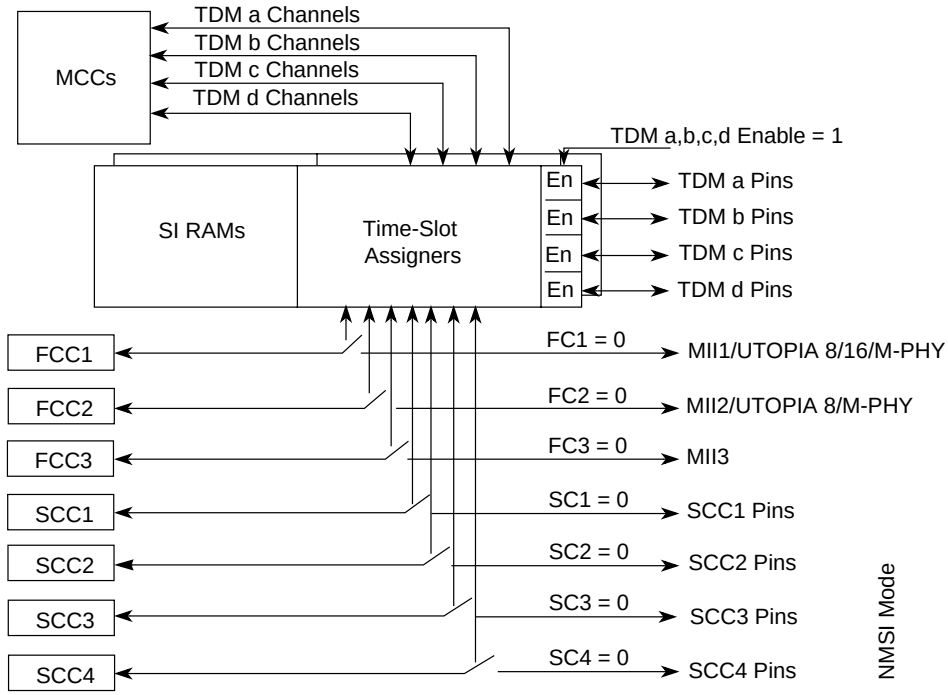


Figure 23-2. Enabling Connections to the TSA

23.3 NMSI Configuration

The CMX supports an NMSI mode for each of the FCCs and SCCs. Each serial device is connected independently either to the NMSI or to the TSA using the clock route registers. The user should note, however, that NMSI pins are multiplexed with other functions at the parallel I/O lines. Therefore, if a combination of TDM and NMSI channels are used, consult the MPC8560 pinout to determine which FCC and SCC to connect and where to connect them.

The clocks provided to the FCCs and SCCs are derived from a bank of 8 internal BRGs and 20 external CLK pins; see Figure 23-3. There are two main advantages to the bank-of-clocks approach. First, a serial device is not forced to choose a serial device clock from a predefined pin or BRG; this allows a flexible pinout-mapping strategy. Second, a group of serial receivers and transmitters that needs the same clock rate can share the same pin. This configuration leaves additional pins for other functions and minimizes potential skew between multiple clock sources.

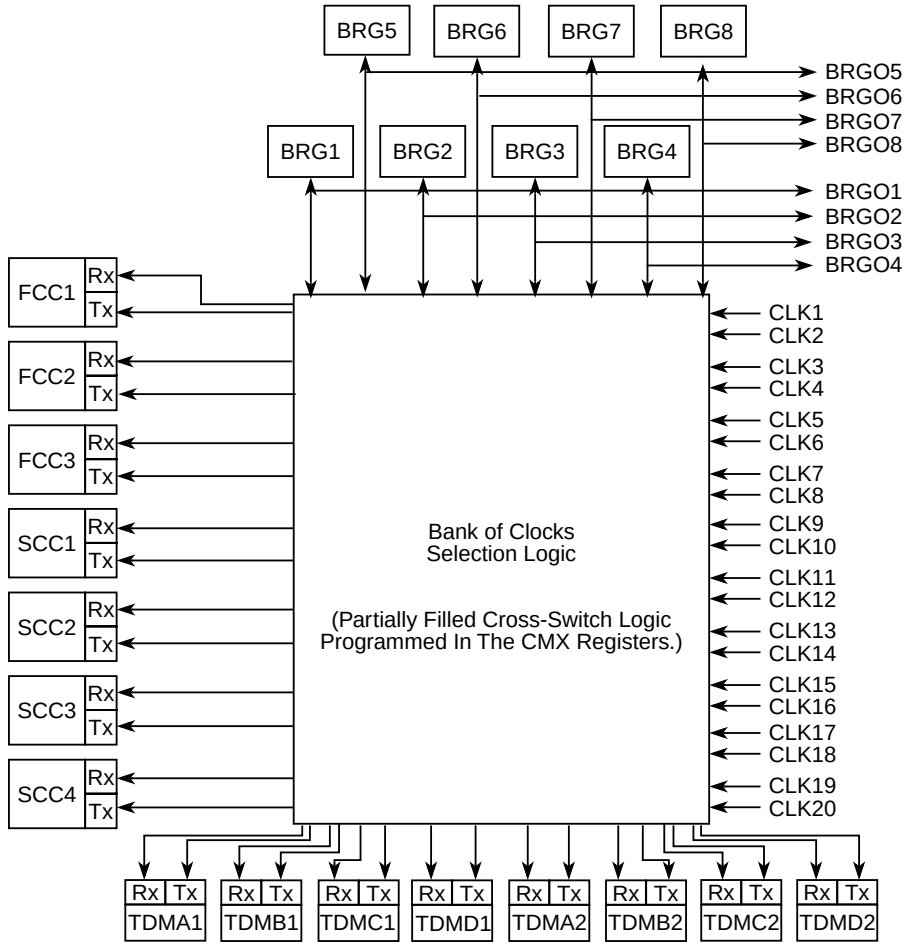


Figure 23-3. Bank of Clocks

The eight BRGs also make their clocks available to external logic, regardless of whether the BRGs are being used by a serial device. Notice that the BRG outputs are multiplexed with other functions; thus, all BRGO*x* pins may not always be available. Chapter 42, “Parallel I/O Ports,” shows the function multiplexing.

Note that in bank-of-clocks mapping only 4 of the 20 sources can be connected to any given FCC or SCC receiver or transmitter.

Table 23-1 shows the clock source options for the serial controllers and TDM channels.

Table 23-1. Clock Source Options

Clock	CLK																				BRG							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8
SCC1 Rx			V	V							V	V									V	V	V	V				
SCC1 Tx			V	V							V	V									V	V	V	V				
SCC2 Rx			V	V							V	V									V	V	V	V				
SCC2 Tx			V	V							V	V									V	V	V	V				
SCC3 Rx					V	V	V	V													V	V	V	V				
SCC3 Tx					V	V	V	V													V	V	V	V				
SCC4 Rx					V	V	V	V													V	V	V	V				
SCC4 Tx					V	V	V	V													V	V	V	V				
FCC1 Rx									V	V	V	V													V	V	V	V
FCC1 Tx									V	V	V	V													V	V	V	V
FCC2 Rx													V	V	V	V									V	V	V	V
FCC2 Tx													V	V	V	V									V	V	V	V
FCC3 Rx													V	V	V	V									V	V	V	V
FCC3 Tx													V	V	V	V									V	V	V	V
TDMA1 Rx	V																		V									
TDMA1 Tx		V																		V								
TDMB1 Rx			V						V																			
TDMB1 Tx				V						V																		
TDMC1 Rx					V								V															
TDMC1 Tx						V								V														
TDMD1 Rx							V								V													
TDMD1 Tx								V								V												
TDMA2 Rx					V								V															
TDMA2 Tx						V								V														
TDMB2 Rx															V		V											
TDMB2 Tx																V		V										
TDMC2 Rx			V														V											
TDMC2 Tx				V														V										
TDMD2 Rx	V																		V									
TDMD2 Tx		V																		V								

Note that after a clock source is selected, the clock is given an internal name. For the FCCs and SCCs, the names are RCLK*x* and TCLK*x*. These internal names are used only in NMSI

mode to specify the clocks sent to the FCCs and SCCs. These names do not correspond to any MPC8560 pins.

23.4 CMX Registers

The following sections describe the CMX registers.

23.4.1 CMX UTOPIA Address Register (CMXUAR)

The CMX UTOPIA address register (CMXUAR), shown in Figure 23-4, defines the connection of FCC1 and FCC2 UTOPIA multiple-PHY addresses to the 20 UTOPIA address pins of the MPC8560; it also defines the connection of a BRG to the FCCs when an internal rate feature is used. This enables the user to implement a multiple-PHY UTOPIA master or slave on both FCC1 and FCC2 using only 20 pins. The user chooses the number of PHYs to use with each interface and the number of address lines needed for each FCC.

	0	1	2	3	4	5	6	7	8	9	10	11	12	15
Field	SAD0	SAD1	SAD2	SAD3	SAD4	—	MAD4	MAD3	F1IRB		F2IRB		—	
Reset	0000_0000_0000_0000													
R/W	R/W													
Addr	0x9_1B0E													

Figure 23-4. CMX UTOPIA Address Register (CMXUAR)

Table 23-2 describes CMXUAR fields.

Table 23-2. CMXUAR Field Descriptions

Bits	Name	Description
0–4	SAD*x*	Slave address input pin *x* connection. Note that the address indexes are relative to FCC1; see Figure 23-7. 0 This address input pin is used by FCC2 in slave mode. 1 This address input pin is used by FCC1 in slave mode.
5	—	Reserved, should be cleared.
6–7	MAD*x*	Master address output pin *x* connection. Note that the address indexes are relative to FCC1; see Figure 23-7. 0 This address output pin is used by FCC2 in master mode. 1 This address output pin is used by FCC1 in master mode.
8–9	F1IRB	FCC1 internal rate BRG selection. Selects the BRG to be connected to FCC 1 for internal rate operation. Used by the ATM controller; see Section 35.2.1.4, “Transmit External Rate and Internal Rate Modes.” 00 FCC1 internal rate clock is BRG5. 01 FCC1 internal rate clock is BRG6. 10 FCC1 internal rate clock is BRG7. 11 FCC1 internal rate clock is BRG8.

Table 23-2. CMXUAR Field Descriptions (continued)

Bits	Name	Description
10–11	F12IRB	FCC2 internal rate BRG selection. Selects the BRG to be connected to FCC 2 for internal rate operation. Used by the ATM controller; see Section 35.2.1.4, “Transmit External Rate and Internal Rate Modes.” 00 FCC2 internal rate clock is BRG5. 01 FCC2 internal rate clock is BRG6. 10 FCC2 internal rate clock is BRG7. 11 FCC2 internal rate clock is BRG8.
12–15	—	Reserved, should be cleared.

Note that each SADx and MADx corresponds to a pair of separate receive and transmit address pins.

The MPC8560 has 16 output address pins and 10 input address pins dedicated for the UTOPIA interface. However, it has two FCCs with two parts each—receiver and transmitter that can be either master or slave concurrently. The MPC8560 allows both FCC1 and FCC2 to connect to the address lines without putting limitations on being a master or slave, as described in the following:

- For master mode: The user has two groups of eight address pins each. Three pins from each group are always connected to FCC1 and three are always connected to FCC2. The user decides which FCC uses the remaining two pins by programming CMXUAR[MAD*x*]. See Figure 23-5.

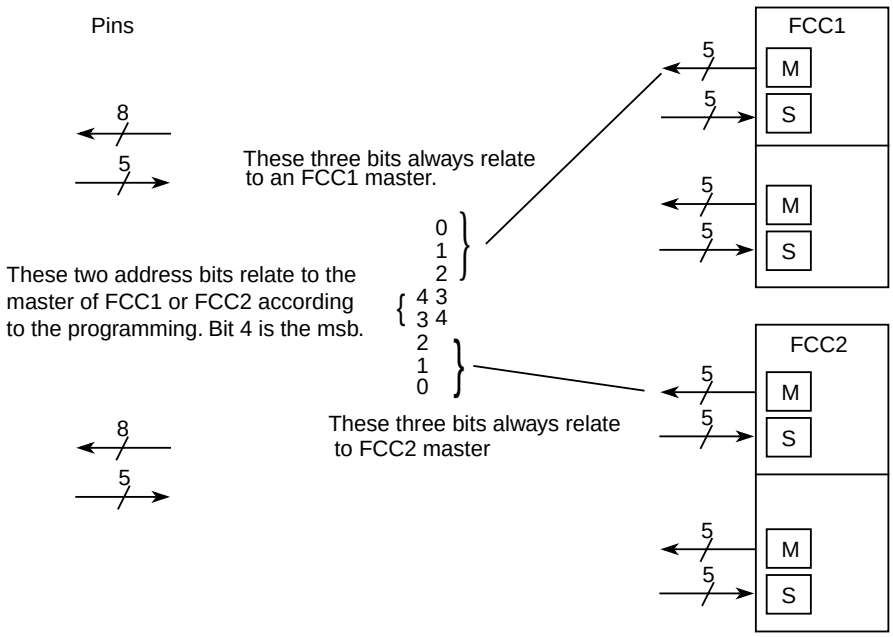


Figure 23-5. Connection of the Master Address

- For slave mode—The user has two groups of five address pins each. The user decides which FCC uses each pin by programming CMXUAR[SAD*x*]. Connect any UTOPIA pins that are not connected to MPC8560 pins to GND. See Figure 23-6.

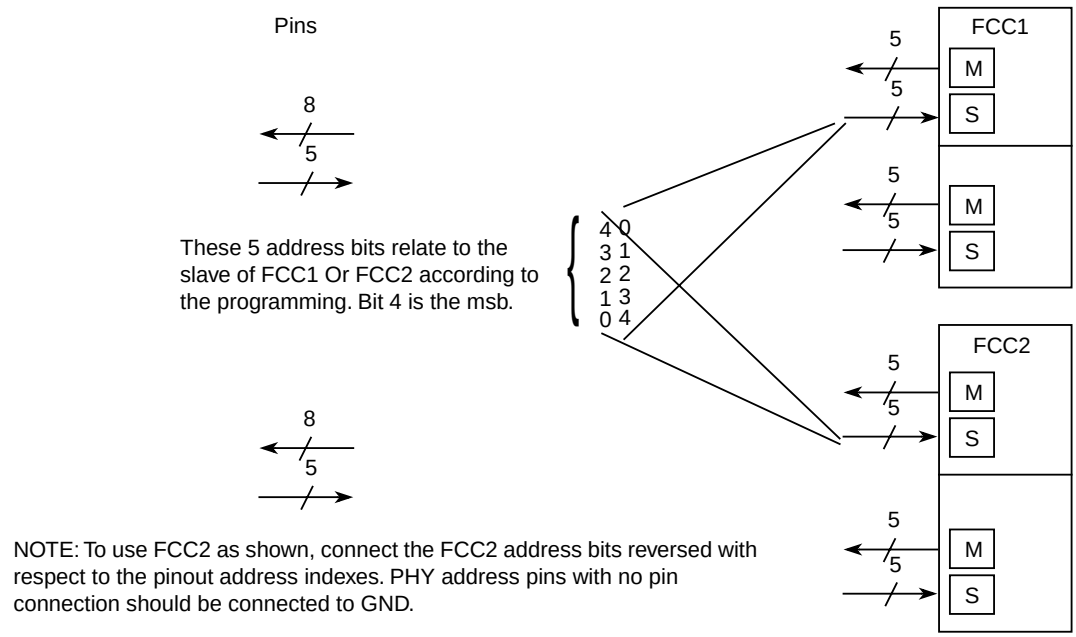


Figure 23-6. Connection of the Slave Address

NOTE

The user must program the addresses of the PHYs to be consecutive for each FCC; that is, the address lines connected to each FCC must be consecutive.

Figure 23-7 describes the interconnection between the receive external multi-PHY bus and the internal FCC1 and FCC2 receive multi-PHY addresses. The same diagram applies to the transmit multi-PHY bus using different dedicated parallel I/O pins.

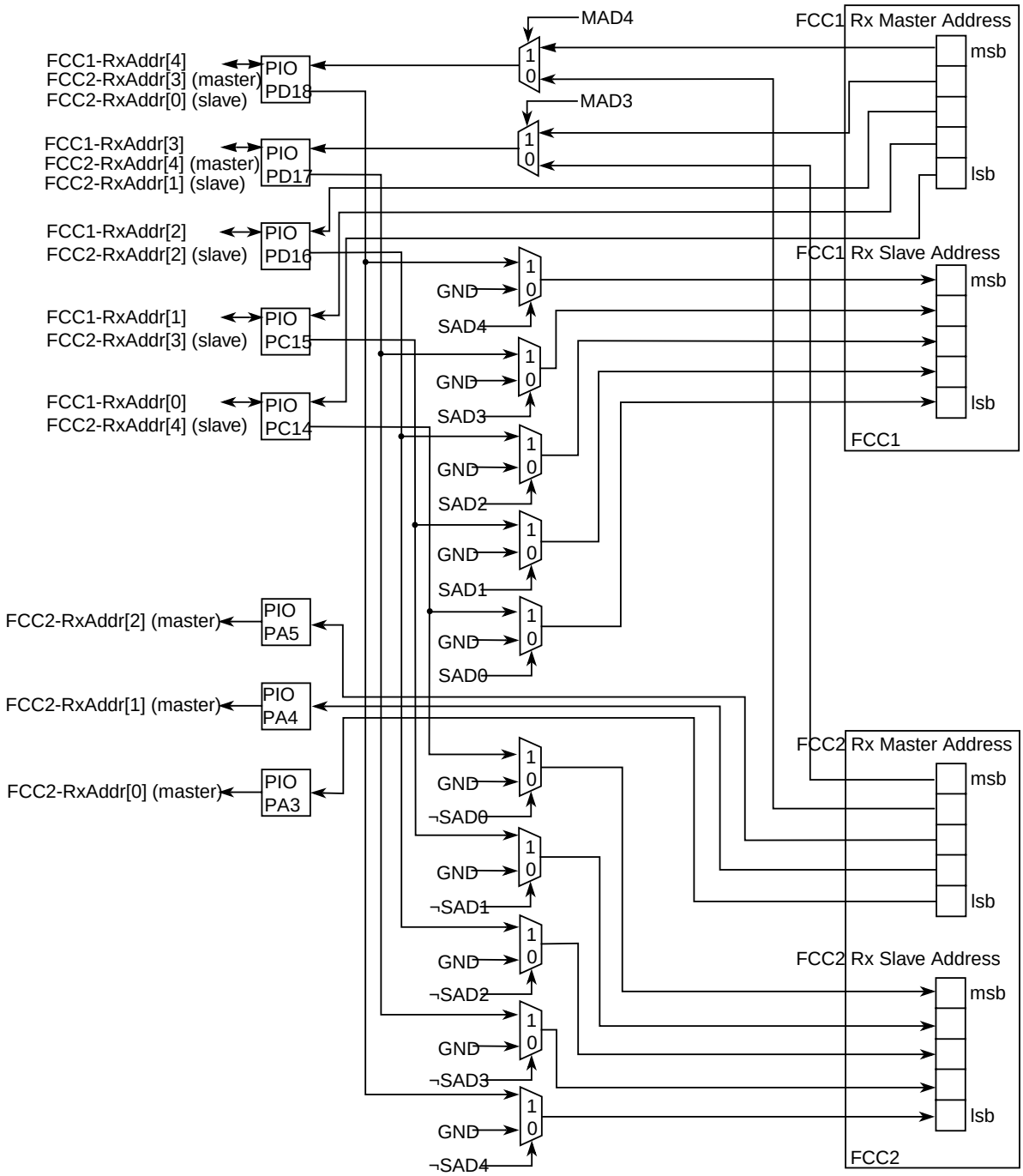


Figure 23-7. Multi-PHY Receive Address Multiplexing

23.4.2 CMX SI1 Clock Route Register (CMXSI1CR)

The CMX SI1 clock route register (CMXSI1CR) defines the connection of SI1 to the clock sources that can be input from the bank of clocks.

	0	1	2	3	4	5	6	7
Field	RTA1CS	RTB1CS	RTC1CS	RTD1CS	TTA1CS	TTB1CS	TTC1CS	TTD1CS
Reset	0000_0000							
R/W	R/W							
Addr	0x9_1B00							

Figure 23-8. CMX SI1 Clock Route Register (CMXSI1CR)

Table 23-3 describes CMXSI1CR fields.

Table 23-3. CMXSI1CR Field Descriptions

Bits	Name	Description
0	RTA1CS	Receive TDM A1 clock source 0 TDM A1 receive clock is CLK1. 1 TDM A1 receive clock is CLK19.
1	RTB1CS	Receive TDM B1 clock source 0 TDM B1 receive clock is CLK3. 1 TDM B1 receive clock is CLK9.
2	RTC1CS	Receive TDM C1 clock source 0 TDM C1 receive clock is CLK5. 1 TDM C1 receive clock is CLK13.
3	RTD1CS	Receive TDM D1 clock source 0 TDM D1 receive clock is CLK7. 1 TDM D1 receive clock is CLK15.
4	TTA1CS	Transmit TDM A1 clock source 0 TDM A1 transmit clock is CLK2. 1 TDM A1 transmit clock is CLK20.
5	TTB1CS	Transmit TDM B1 clock source 0 TDM B1 transmit clock is CLK4. 1 TDM B1 transmit clock is CLK10.
6	TTC1CS	Transmit TDM C1 clock source 0 TDM C1 transmit clock is CLK6. 1 TDM C1 transmit clock is CLK14.
7	TTD1CS	Transmit TDM D1 clock source 0 TDM D1 transmit clock is CLK8. 1 TDM D1 transmit clock is CLK16.

23.4.3 CMX SI2 Clock Route Register (CMXSI2CR)

The CMX SI2 clock route register (CMXSI2CR) defines the connection of SI2 to the clock sources that can be input from the bank of clocks.

	0	1	2	3	4	5	6	7
Field	RTA2CS	RTB2CS	RTC2CS	RTD2CS	TTA2CS	TTB2CS	TTC2CS	TTD2CS
Reset	0000_0000							
R/W	R/W							
Addr	0x9_1B02							

Figure 23-9. CMX SI2 Clock Route Register (CMXSI2CR)

Table 23-4 describes CMXSI2CR fields.

Table 23-4. CMXSI2CR Field Descriptions

Bits	Name	Description
0	RTA2CS	Receive TDM A2 clock source 0 TDM A2 receive clock is CLK13. 1 TDM A2 receive clock is CLK5.
1	RTB2CS	Receive TDM B2 clock source 0 TDM B2 receive clock is CLK15. 1 TDM B2 receive clock is CLK17.
2	RTC2CS	Receive TDM C2 clock source 0 TDM C2 receive clock is CLK3. 1 TDM C2 receive clock is CLK17.
3	RTD2CS	Receive TDM D2 clock source 0 TDM D2 receive clock is CLK1. 1 TDM D2 receive clock is CLK19.
4	TTA2CS	Transmit TDM A2 clock source 0 TDM A2 transmit clock is CLK14. 1 TDM A2 transmit clock is CLK6.
5	TTB2CS	Transmit TDM B2 clock source 0 TDM B2 transmit clock is CLK16. 1 TDM B2 transmit clock is CLK18.
6	TTC2CS	Transmit TDM C2 clock source 0 TDM C2 transmit clock is CLK4. 1 TDM C2 transmit clock is CLK18.
7	TTD2CS	Transmit TDM D2 clock source 0 TDM D2 transmit clock is CLK2. 1 TDM D2 transmit clock is CLK20.

23.4.4 CMX FCC Clock Route Register (CMXFCR)

The CMX FCC clock route register (CMXFCR) defines the connection of the FCCs to the TSA and to the clock sources from the bank of clocks.

	0	1	2–4	5–7	8	9	10–12	13–15
Field	—	FC1	RF1CS	TF1CS	—	FC2	RF2CS	TF2CS
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x9_1B04							

	16	17	18–20	21–23	24–31
Field	—	FC3	RF3CS	TF3CS	—
Reset	0000_0000_0000_0000				
R/W	R/W				
Addr	0x9_1B06				

Figure 23-10. CMX FCC Clock Route Register (CMXFCR)

Table 23-5 describes CMXFCR fields.

Table 23-5. CMXFCR Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared
1	FC1	Defines the FCC1 connection 0 FCC1 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus FCCn pins is made in the parallel I/O control register. 1 FCC1 is connected to the TSA of the SIs. The NMSIx pins are available for other purposes.
2–4	RF1CS	Receive FCC1 clock source (NMSI mode). Ignored if FCC1 is connected to the TSA (FC1 = 1). 000 FCC1 receive clock is BRG5. 001 FCC1 receive clock is BRG6. 010 FCC1 receive clock is BRG7. 011 FCC1 receive clock is BRG8. 100 FCC1 receive clock is CLK9. 101 FCC1 receive clock is CLK10. 110 FCC1 receive clock is CLK11. 111 FCC1 receive clock is CLK12.
5–7	TF1CS	Transmit FCC1 clock source (NMSI mode). Ignored if FCC1 is connected to the TSA (FC1 = 1). 000 FCC1 transmit clock is BRG5. 001 FCC1 transmit clock is BRG6. 010 FCC1 transmit clock is BRG7. 011 FCC1 transmit clock is BRG8. 100 FCC1 transmit clock is CLK9. 101 FCC1 transmit clock is CLK10. 110 FCC1 transmit clock is CLK11. 111 FCC1 transmit clock is CLK12.
8	—	Reserved, should be cleared
9	FC2	Defines the FCC2 connection 0 FCC2 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus FCCn pins is made in the parallel I/O control register. 1 FCC2 is connected to the TSA of the SIs. The NMSIx pins are available for other purposes.

Table 23-5. CMXFCR Field Descriptions (continued)

Bits	Name	Description
10–12	RF2CS	Receive FCC2 clock source (NMSI mode). Ignored if FCC2 is connected to the TSA (FC2 = 1). 000 FCC2 receive clock is BRG5. 001 FCC2 receive clock is BRG6. 010 FCC2 receive clock is BRG7. 011 FCC2 receive clock is BRG8. 100 FCC2 receive clock is CLK13. 101 FCC2 receive clock is CLK14. 110 FCC2 receive clock is CLK15. 111 FCC2 receive clock is CLK16.
13–15	TF2CS	Transmit FCC2 clock source (NMSI mode). Ignored if FCC2 is connected to the TSA (FC2 = 1). 000 FCC2 transmit clock is BRG5. 001 FCC2 transmit clock is BRG6. 010 FCC2 transmit clock is BRG7. 011 FCC2 transmit clock is BRG8. 100 FCC2 transmit clock is CLK13. 101 FCC2 transmit clock is CLK14. 110 FCC2 transmit clock is CLK15. 111 FCC2 transmit clock is CLK16.
16	—	Reserved, should be cleared
17	FC3	Defines the FCC3 connection 0 FCC3 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus FCCn pins is made in the parallel I/O control register. 1 FCC3 is connected to the TSA of the SIs. The NMSIx pins are available for other purposes.
18–20	RF3CS	Receive FCC3 clock source (NMSI mode). Ignored if FCC3 is connected to the TSA (FC3 = 1). 000 FCC3 receive clock is BRG5. 001 FCC3 receive clock is BRG6. 010 FCC3 receive clock is BRG7. 011 FCC3 receive clock is BRG8. 100 FCC3 receive clock is CLK13. 101 FCC3 receive clock is CLK14. 110 FCC3 receive clock is CLK15. 111 FCC3 receive clock is CLK16.
21–23	TF3CS	Transmit FCC3 clock source (NMSI mode). Ignored if FCC3 is connected to the TSA (FC3 = 1). 000 FCC3 transmit clock is BRG5. 001 FCC3 transmit clock is BRG6. 010 FCC3 transmit clock is BRG7. 011 FCC3 transmit clock is BRG8. 100 FCC3 transmit clock is CLK13. 101 FCC3 transmit clock is CLK14. 110 FCC3 transmit clock is CLK15. 111 FCC3 transmit clock is CLK16.
24–31	—	Reserved, should be cleared

23.4.5 CMX SCC Clock Route Register (CMXSCR)

The CMX SCC clock route register (CMXSCR) defines the connection of the SCCs to the TSA and to the clock sources from the bank of clocks. This register also enables the use of the external grant pin.

	0	1	2 4	5 7	8	9	10 12	13 15
Field	GR1	SC1	RS1CS	TS1CS	GR2	SC2	RS2CS	TS2CS
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x9_1B08							

	16	17	18 20	21 23	24	25	26 28	29 31
Field	GR3	SC3	RS3CS	TS3CS	GR4	SC4	RS4CS	TS4CS
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x9_1B0A							

Figure 23-11. CMX SCC Clock Route Register (CMXSCR)

Table 23-6 describes CMXSCR fields.

Table 23-6. CMXSCR Field Descriptions

Bits	Name	Description
0	GR1	Grant support of SCC1 0 SCC1 transmitter does not support the grant mechanism. The grant is always asserted internally. 1 SCC1 transmitter supports the grant mechanism as determined by the GMx bit of a serial device channel.
1	SC1	SCC1 connection 0 SCC1 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus SCCn pins is made in the parallel I/O control register. 1 SCC1 is connected to TSA of the SIs. The NMSIx pins are available for other purposes.
2–4	RS1CS	Receive SCC1 clock source (NMSI mode). Ignored if SCC1 is connected to the TSA (SC1 = 1). 000 SCC1 receive clock is BRG1. 001 SCC1 receive clock is BRG2. 010 SCC1 receive clock is BRG3. 011 SCC1 receive clock is BRG4. 100 SCC1 receive clock is CLK11. 101 SCC1 receive clock is CLK12. 110 SCC1 receive clock is CLK3. 111 SCC1 receive clock is CLK4.

Table 23-6. CMXSCR Field Descriptions (continued)

Bits	Name	Description
5–7	TS1CS	Transmit SCC1 clock source (NMSI mode). Ignored if SCC1 is connected to the TSA (SC1 = 1). 000 SCC1 transmit clock is BRG1. 001 SCC1 transmit clock is BRG2. 010 SCC1 transmit clock is BRG3. 011 SCC1 transmit clock is BRG4. 100 SCC1 transmit clock is CLK11. 101 SCC1 transmit clock is CLK12. 110 SCC1 transmit clock is CLK3. 111 SCC1 transmit clock is CLK4.
8	GR2	Grant support of SCC2 0 SCC2 transmitter does not support the grant mechanism. The grant is always asserted internally. 1 SCC2 transmitter supports the grant mechanism as determined by the GMx bit of a serial device channel.
9	SC2	SCC2 connection 0 SCC2 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus SCCn pins is made in the parallel I/O control register. 1 SCC2 is connected to TSA of the SIs. The NMSIx pins are available for other purposes.
10–12	RS2CS	Receive SCC2 clock source (NMSI mode). Ignored if SCC2 is connected to the TSA (SC2 = 1). 000 SCC2 receive clock is BRG1. 001 SCC2 receive clock is BRG2. 010 SCC2 receive clock is BRG3. 011 SCC2 receive clock is BRG4. 100 SCC2 receive clock is CLK11. 101 SCC2 receive clock is CLK12. 110 SCC2 receive clock is CLK3. 111 SCC2 receive clock is CLK4.
13–15	TS2CS	Transmit SCC2 clock source (NMSI mode). Ignored if SCC2 is connected to the TSA (SC2 = 1). 000 SCC2 transmit clock is BRG1. 001 SCC2 transmit clock is BRG2. 010 SCC2 transmit clock is BRG3. 011 SCC2 transmit clock is BRG4. 100 SCC2 transmit clock is CLK11. 101 SCC2 transmit clock is CLK12. 110 SCC2 transmit clock is CLK3. 111 SCC2 transmit clock is CLK4.
16	GR3	Grant support of SCC3 0 SCC3 transmitter does not support the grant mechanism. The grant is always asserted internally. 1 SCC3 transmitter supports the grant mechanism as determined by the GMx bit of a serial device channel.
17	SC3	SCC3 connection 0 SCC3 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus SCCn pins is made in the parallel I/O control register. 1 SCC3 is connected to TSA of the SIs. The NMSIx pins are available for other purposes.

Table 23-6. CMXSCR Field Descriptions (continued)

Bits	Name	Description
18–20	RS3CS	Receive SCC3 clock source (NMSI mode). Ignored if SCC3 is connected to the TSA (SC3 = 1). 000 SCC3 receive clock is BRG1. 001 SCC3 receive clock is BRG2. 010 SCC3 receive clock is BRG3. 011 SCC3 receive clock is BRG4. 100 SCC3 receive clock is CLK5. 101 SCC3 receive clock is CLK6. 110 SCC3 receive clock is CLK7. 111 SCC3 receive clock is CLK8.
21–23	TS3CS	Transmit SCC3 clock source (NMSI mode). Ignored if SCC3 is connected to the TSA (SC3 = 1). 000 SCC3 transmit clock is BRG1. 001 SCC3 transmit clock is BRG2. 010 SCC3 transmit clock is BRG3. 011 SCC3 transmit clock is BRG4. 100 SCC3 transmit clock is CLK5. 101 SCC3 transmit clock is CLK6. 110 SCC3 transmit clock is CLK7. 111 SCC3 transmit clock is CLK8.
24	GR4	Grant support of SCC4 0 SCC4 transmitter does not support the grant mechanism. The grant is always asserted internally. 1 SCC4 transmitter supports the grant mechanism as determined by the GMx bit of a serial device channel.
25	SC4	SCC4 connection 0 SCC4 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus SCCn pins is made in the parallel I/O control register. 1 SCC4 is connected to TSA of the SIs. The NMSIx pins are available for other purposes.
26–28	RS4CS	Receive SCC4 clock source (NMSI mode). Ignored if SCC4 is connected to the TSA (SC4 = 1). 000 SCC4 receive clock is BRG1. 001 SCC4 receive clock is BRG2. 010 SCC4 receive clock is BRG3. 011 SCC4 receive clock is BRG4. 100 SCC4 receive clock is CLK5. 101 SCC4 receive clock is CLK6. 110 SCC4 receive clock is CLK7. 111 SCC4 receive clock is CLK8
29–31	TS4CS	Transmit SCC4 clock source (NMSI mode). Ignored if SCC4 is connected to the TSA (SC4 = 1). 000 SCC4 transmit clock is BRG1. 001 SCC4 transmit clock is BRG2. 010 SCC4 transmit clock is BRG3. 011 SCC4 transmit clock is BRG4. 100 SCC4 transmit clock is CLK5. 101 SCC4 transmit clock is CLK6. 110 SCC4 transmit clock is CLK7. 111 SCC4 transmit clock is CLK8

Chapter 24
Baud-Rate Generators (BRGs)

The CPM contains eight independent, identical baud-rate generators (BRGs) that can be used with the FCCs and SCCs. The clocks produced by the BRGs are sent to the bank-of-clocks selection logic, where they can be routed to the controllers. In addition, the output of a BRG can be routed to a pin to be used externally. The following is a list of BRGs' main features:

- Eight independent and identical BRGs
- On-the-fly changes allowed
- Each BRG can be routed to one or more FCCs or SCCs
- A 16x divider option allows slow baud rates at high system frequencies
- Each BRG contains an autobaud support option
- Each BRG output can be routed to a pin (BRGO*n*)

Figure 24-1 shows a BRG.

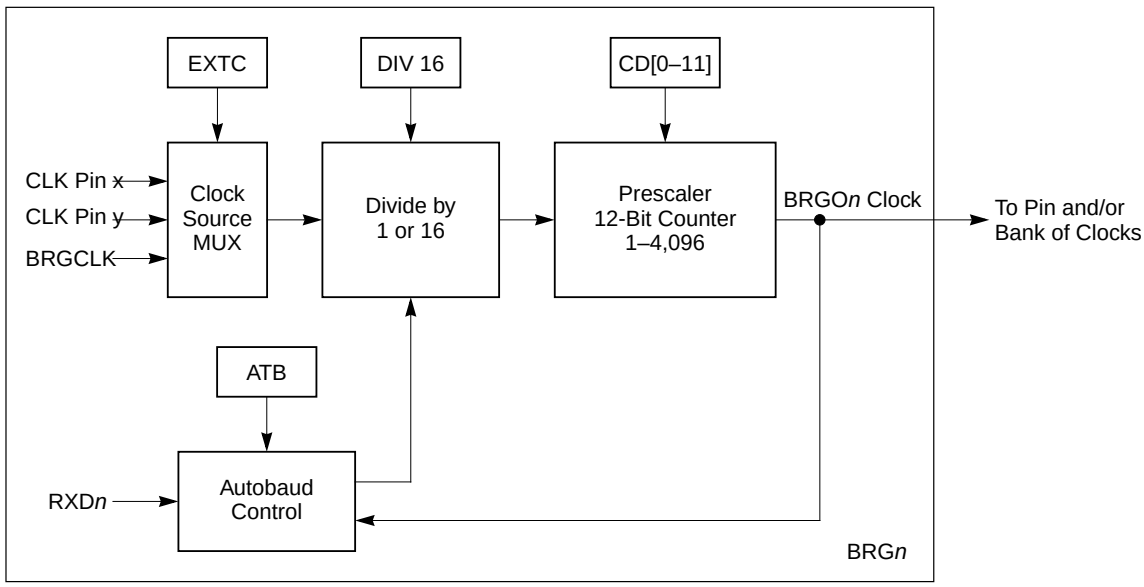


Figure 24-1. Baud-Rate Generator (BRG) Block Diagram

Each BRG clock source can be BRGCLK, or a choice of two external clocks (selected in BRGC*x*[EXTC]). The BRGCLK is an internal signal generated in the MPC8560 clock synthesizer specifically for the BRGs, the SPI, and the I^2C internal BRG. Alternatively, external clock pins can be configured as clock sources. The external source option allows flexible baud-rate frequency generation, independent of the system frequency. Additionally, the external source option allows a single external frequency to be the source for multiple BRGs. The external source signals are not synchronized internally before being used by the BRG.

The BRG provides a divide-by-16 option (BRGC*x*[DIV16]) and a 12-bit prescaler (BRGC*x*[CD]) to divide the source clock frequency. The combined source-clock divide factor can be changed on-the-fly; however, two changes should not occur within two source clock periods.

The prescaler output is sent internally to the bank of clocks and can also be output externally on BRGO*n* through the parallel I/O ports. If the BRG divides the clock by an even value, the transitions of BRGO*n* always occur on the falling edge of the source clock. If the divide factor is odd, the transitions alternate between the falling and rising edges of the source clock. Additionally, the output of the BRG can be sent to the autobaud control block.

24.1 System Clock Control Register (SCCR)

The system clock control register (SCCR), shown in Figure 24-2, is memory-mapped into the MPC8560's internal space. The SCCR defines the BRGCLK frequency to be supplied to the BRGs.

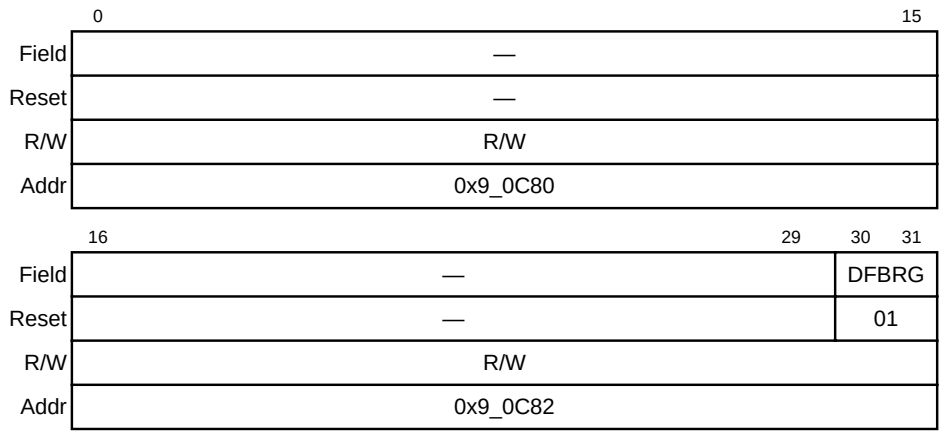


Figure 24-2. System Clock Control Register (SCCR)

Table 24-1 describes SCCR fields.

Table 24-1. SCCR Field Descriptions

Bits	Name	Defaults		Description
		POR	Hard Reset	
0–29	—			Reserved
30–31	DFBRG	01	Unaffected	Division factor of BRG_CLK relative to VCO_OUT (which is twice the CPM clock). Defines the BRG_CLK frequency. Changing the value does not result in a loss of lock condition. 00 Divide by 4 01 Divide by 16 (normal operation) 10 Divide by 64 11 Divide by 256

24.2 BRG Configuration Registers 1–8 (BRGC*x*)

The BRG configuration registers (BRGC*x*) are shown in Figure 24-3. A reset disables the BRG and drives the BRGO output clock high. The BRGC can be written at any time with no need to disable the SCCs or external devices that are connected to BRGO. Configuration changes occur at the end of the next BRG clock cycle (no spikes occur on the BRGO output clock). BRGC can be changed on-the-fly; however, two changes should not occur within a time equal to two source clock periods.

	0 – 13	14	15
Field	—	RST	EN
Reset	0000_0000_0000_0000		
R/W	R/W		
Addr	0x9_19F0 (BRGC1), 0x9_19F4 (BRGC2), 0x9_19F8 (BRGC3), 0x9_19FC (BRGC4), 0x9_15F0 (BRGC5), 0x9_15F4 (BRGC6), 0x9_15F8 (BRGC7), 0x9_15FC (BRGC8)		

	16 – 17	18	19 – 30	31
Field	EXTC	ATB	CD	DIV16
Reset	0000_0000_0000_0000			
R/W	R/W			
Addr	0x9_19F2 (BRGC1), 0x9_19F6 (BRGC2), 0x9_19FA (BRGC3), 0x9_19FE (BRGC4), 0x9_15F2 (BRGC5), 0x9_15F6 (BRGC6), 0x9_15FA (BRGC7), 0x9_15FE (BRGC8)			

Figure 24-3. Baud-Rate Generator Configuration Registers (BRGC*x*)

Table 24-2 describes the BRGC*x* fields.

Table 24-2. BRGC*x* Field Descriptions

Bits	Name	Description
0–13	—	Reserved, should be cleared.
14	RST	Reset BRG. Performs a software reset of the BRG identical to that of an external reset. A reset disables the BRG and drives BRGO high. This is externally visible only if BRGO is connected to the corresponding parallel I/O pin. 0 Enable the BRG. 1 Reset the BRG (software reset).
15	EN	Enable BRG count. Used to dynamically stop the BRG from counting—useful for low-power modes. 0 Stop all clocks to the BRG. 1 Enable clocks to the BRG.
16–17	EXTC	External clock source. Selects the BRG input clock. See Table 24-3. 00 The BRG input clock comes from the BRGCLK (internal clock generated from the CPM clock); see Section 9.8, “System Clock Control Register (SCCR).” 01 If BRG1, 2, 5, 6: The BRG input clock comes from the CLK3 pin. If BRG3, 4, 7, 8: The BRG input clock comes from the CLK9 pin 10 If BRG1, 2, 5, 6: The BRG input clock comes from the CLK5 pin. If BRG3, 4, 7, 8: The BRG input clock comes from the CLK15 pin 11 Reserved
18	ATB	Autobaud. Selects autobaud operation of the BRG on the corresponding RXD. ATB must remain zero until the SCC receives the three Rx clocks. Then the user must set ATB to obtain the correct baud rate. After the baud rate is obtained and locked, it is indicated by setting AB in the UART event register. 0 Normal operation of the BRG. 1 When RXD goes low, the BRG determines the length of the start bit and synchronizes the BRG to the actual baud rate.
19–30	CD	Clock divider. CD presets an internal 12-bit counter that is decremented at the DIV16 output rate. When the counter reaches zero, it is reloaded with CD. CD = 0xFFF produces the minimum clock rate for BGRO (divide by 4,096); CD = 0x000 produces the maximum rate (divide by 1). When dividing by an odd number, the counter ensures a 50% duty cycle by asserting the terminal count once on clock low and next on clock high. The terminal count signals counter expiration and toggles the clock. See Section 24.4, “UART Baud Rate Examples.”
31	DIV16	Divide-by-16. Selects a divide-by-1 or divide-by-16 prescaler before reaching the clock divider. See Section 24.4, “UART Baud Rate Examples.” 0 Divide by 1. 1 Divide by 16.

Table 24-3 shows the possible external clock sources for the BRGs.

Table 24-3. BRG External Clock Source Options

BRG	CLK																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
BRG1			V		V															
BRG2			V		V															
BRG3									V						V					
BRG4									V						V					
BRG5			V		V															
BRG6			V		V															
BRG7									V						V					
BRG8									V						V					

24.3 Autobaud Operation on a UART

During the autobaud process, a UART deduces the baud rate of its received character stream by examining the received pattern and its timing. A built-in autobaud control function automatically measures the length of a start bit and modifies the baud rate accordingly.

If the autobaud bit BRGC*x*[ATB] is set, the autobaud control function starts searching for a low level on the corresponding RXD*n* input, which it assumes marks the beginning of a start bit, and begins counting the start bit length. During this time, the BRG output clock toggles for 16 BRG clock cycles at the BRG source clock rate and then stops with BRGO*n* in the low state.

When RXD*n* goes high again, the autobaud control block rewrites BRGC*x*[CD, DIV16] to the divide ratio found, which at high baud rates may not be exactly the final rate desired (for example, 56,600 may result rather than 57,600). An interrupt can be enabled in the UART SCC event register to report that the autobaud controller rewrote BRGC*x*. The interrupt handler can then adjust BRGC*x*[CD, DIV16] (see Table 24-4) for accuracy before the first character is fully received, ensuring that the UART recognizes all characters.

After a full character is received, the software can verify that the character matches a predefined value (such as ‘a’ or ‘A’). Software should then check for other characters (such as ‘t’ or ‘T’) and program the preferred parity mode in the UART’s protocol-specific mode register (PSMR).

Note that the SCC associated with this BRG must be programmed to UART mode and select the 16× option for TDCR and RDCR in the general SCC mode register low. Input frequencies such as 1.8432, 3.68, 7.36, and 14.72 MHz should be used. The SCC performing the autobaud function must be connected to that SCC’s BRG; that is, SCC2 must be clocked by BRG2, and so on.

Also, to detect an autobaud lock and generate an interrupt, the SCC must receive three full Rx clocks from the BRG before the autobaud process begins. To do this, first clear BRGC*x*[ATB] and enable the BRG Rx clock to the highest frequency. Then, immediately before the autobaud process starts (after device initialization), set BRGC*x*[ATB].

24.4 UART Baud Rate Examples

For synchronous communication using the internal BRG, the BRGO must not exceed the BRG input clock divided by 2. Therefore, with a BRG input clock of 66MHz (generated using an external clock source: refer to BRGCx[EXTC]), the maximum BRGO rate is 33MHz. Program the UART to 16× oversampling when using the SCC as a UART. Rates of 8× and 32× are also available. Assuming 16× oversampling is chosen in the UART, the maximum data rate is 66 MHz ÷ 16 = 4.125 Mbps. Keeping the above in mind, use the following formula to calculate the bit rate based on a particular BRG configuration for a UART:

$$\text{Async Baud Rate} = \frac{\text{BRGCLK or External Clock Source}}{(\text{Prescale Divider}) \sum (\text{Clock Divider} + 1) \sum (\text{Sampling Rate})}$$

$$= \frac{\text{BRGCx[EXTC]}}{(\text{BRGCx[DIV16]}) \sum (\text{BRGCx[CD]} + 1) \sum (\text{GSMRx_L[xDCR]})}$$

Table 24-4 lists typical bit rates of asynchronous communication. Note that here the internal clock rate is assumed to be 16× the baud rate; that is, GSMR*x*_L[TDCR] = GSMR*x*_L[RDCR] = 0b10.

Table 24-4. Typical Baud Rates for Asynchronous Communication

Baud Rate	Using a 66-MHz BRG Input Clock		
	BRGC*x*[DIV16]	BRGC*x*[CD]	Actual Frequency (Hz)
75	1	3436	75.01
150	1	1718	149.98
300	1	858	300.13
600	1	429	599.56
1200	0	3436	1200.2
2400	0	1718	2399.7
4800	0	858	4802.1
9600	0	429	9593.0
19,200	0	214	19,186
38,400	0	106	38,551
57,600	0	71	57,292

Table 24-4. Typical Baud Rates for Asynchronous Communication (continued)

Baud Rate	Using a 66-MHz BRG Input Clock		
	BRGC*x*[DIV16]	BRGC*x*[CD]	Actual Frequency (Hz)
115,200	0	35	114,583
460,000	0	8	458,333

For synchronous communication, the internal clock is identical to the baud-rate output. To get the preferred rate, select the system clock according to the following:

$$\text{Sync Baud Rate} = \frac{\text{BRGCLK or External Clock Source}}{(\text{Prescale Divider}) \sum (\text{Clock Divider} + 1)}$$

$$= \frac{\text{BRGCx[EXTC]}}{(\text{BRGCx[DIV16]}) \sum (\text{BRGCx[CD]} + 1)}$$

For example, to get a rate of 64 kbps, the system clock can be 24.96 MHz, BRGC*x*[DIV16] = 0, and BRGC*x*[CD] = 389.

Chapter 25
CPM Timers

The CPM includes four identical 16-bit general-purpose timers or two 32-bit timers. Each general-purpose timer consists of a timer mode register (TMR), a timer capture register (TCR), a timer counter (TCN), a timer reference register (TRR), a timer event register (TER), and a timer global configuration register (TGCR). The TMRs contain the prescaler values programmed by the user.

Figure 25-1 shows the timer block diagram.

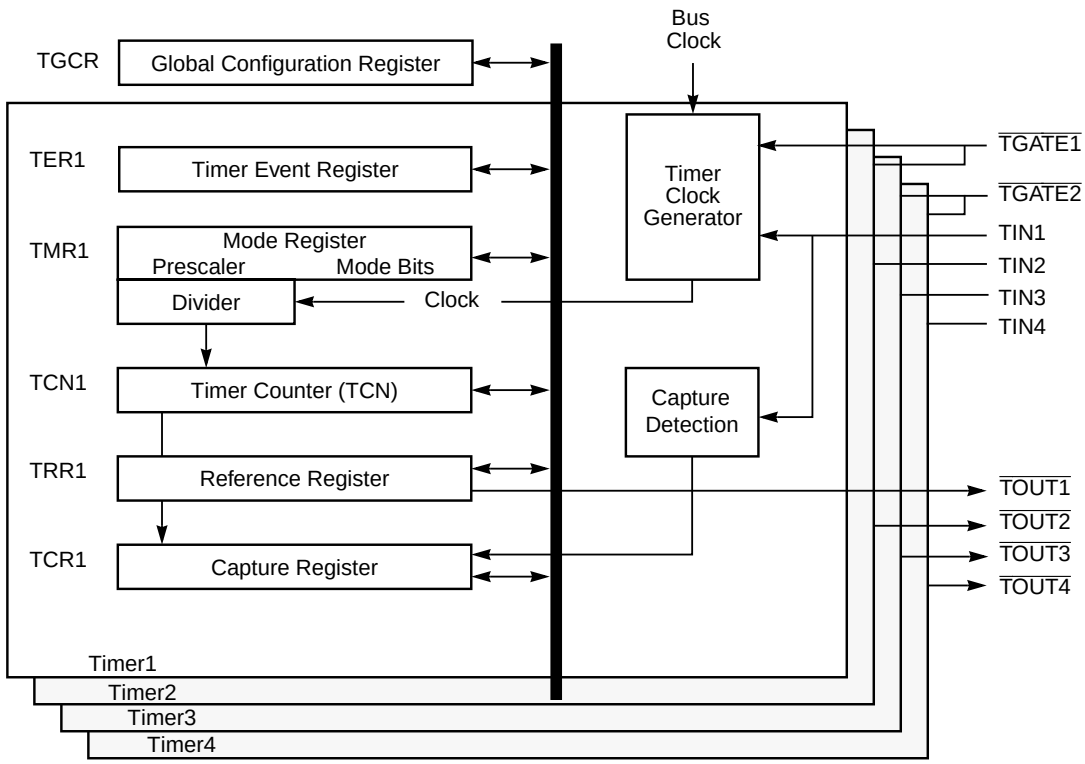


Figure 25-1. Timer Block Diagram

Pin assignments for TIN*x*, $\overline{\text{TGATE}x}$, and $\overline{\text{TOUT}x}$ are described in Section 42.5, “Ports Tables.”

25.1 Features

The key features of the timer include the following:

- The maximum input clock is the bus clock
- Maximum period of 4 seconds (at 66 MHz)
- 16-nanosecond resolution (at 66 MHz)
- Programmable sources for the clock input
- Input capture capability
- Output compare with programmable mode for the output pin
- Two timers cascade internally or externally to form a 32-bit timer
- Free run and restart modes
- Functional compatibility with timers on the MC68360, MPC860, and MPC8260

25.2 General-Purpose Timer Units

The clock input to the prescaler can be selected from the following three sources:

- Bus clock (CLKIN)
- Bus clock divided by 16 (CLKIN/16)
- Corresponding TIN*x*, programmed in the parallel port registers

The bus clock is generated in the clock synthesizer and defaults to the bus frequency. However, the bus clock has the option to be divided before it leaves the clock synthesizer. This mode, called slow go, is used to save power. Whatever the resulting frequency of the bus clock, the user can either choose that frequency or the frequency divided by 16 as the input to the prescaler of each timer. Alternatively, the user may prefer TIN*x* to be the clock source. TIN*x* is internally synchronized to the internal clock. If the user has chosen to internally cascade two 16-bit timers to a 32-bit timer, then a timer can use the clock generated by the output of another timer.

The clock input source is selected by the corresponding TMR[ICLK] bits. The prescaler is programmed to divide the clock input by values from 1 to 256 and the output of the prescaler is used as an input to the 16-bit counter. The best resolution of the timer is one clock cycle (16 ns at 66 MHz). The maximum period (when the reference value is all ones) is 268,435,456 cycles (4 seconds at 66 MHz).

Each timer can be configured to count until a reference is reached and then either begin a new time count immediately or continue to run. The FRR bit of the corresponding TMR selects each mode. Upon reaching the reference value, the corresponding TER bit is set and an interrupt is issued if TMR[ORI] = 1. The timers can output a signal on the timer outputs ($\overline{\text{TOUT1}}$–$\overline{\text{TOUT4}}$) when the reference value is reached (selected by the corresponding TMR[OM]). This signal can be an active-low pulse or a toggle of the current output. The

output can also be connected internally to the input of another timer, resulting in a 32-bit timer.

In addition, each timer has a 16-bit TCR used to latch the value of the counter when a defined transition of TIN1, TIN2, TIN3, or TIN4 is sensed by the corresponding input capture edge detector. The type of transition triggering the capture is selected by the corresponding TMR[CE] bits. Upon a capture or reference event, the corresponding TER bit is set and a maskable interrupt request is issued to the interrupt controller. The timers may be gated/restarted by an external gate signal. There are two gate signals—$\overline{\text{TGATE1}}$ controls timer 1 and/or 2 and $\overline{\text{TGATE2}}$ controls timer 3 and/or 4. Normal gate mode enables the count on a falling edge of $\overline{\text{TGATE}x}$ and disables the count on the rising edge of $\overline{\text{TGATE}x}$. This mode allows the timer to count conditionally, based on the state of $\overline{\text{TGATE}x}$.

The restart gate mode performs the same function as normal mode, except it also resets the counter on the falling edge of $\overline{\text{TGATE}x}$. This mode has applications in pulse interval measurement and bus monitoring as follows:

- Pulse measurement—The restart gate mode can measure a low $\overline{\text{TGATE}x}$. The rising edge of $\overline{\text{TGATE}x}$ completes the measurement and if $\overline{\text{TGATE}x}$ is connected externally to TIN*x*, it causes the timer to capture the count value and generate a rising-edge interrupt.
- Bus monitoring—The restart gate mode can detect a signal that is abnormally stuck low. The bus signal should be connected to $\overline{\text{TGATE}x}$. The timer count is reset on the falling edge of the bus signal and if the bus signal does not go high again within the number of user-defined clocks, an interrupt can be generated.

The gate function is enabled in the TMR; the gate operating mode is selected in the TGCR.

NOTE

$\overline{\text{TGATE}x}$ is internally synchronized to the system clock. If $\overline{\text{TGATE}x}$ meets the asynchronous input setup time, the counter begins counting after one system clock when working with the internal clock.

25.2.1 Cascaded Mode

In this mode, two 16-bit timers can be internally cascaded to form a 32-bit counter. Timer 1 may be internally cascaded to timer 2, and timer 3 can be internally cascaded to timer 4. Because the decision to cascade timers is made independently, the user can select two 16-bit timers or one 32-bit timer. TGCR is used to put the timers into cascaded mode, as shown in Figure 25-2.

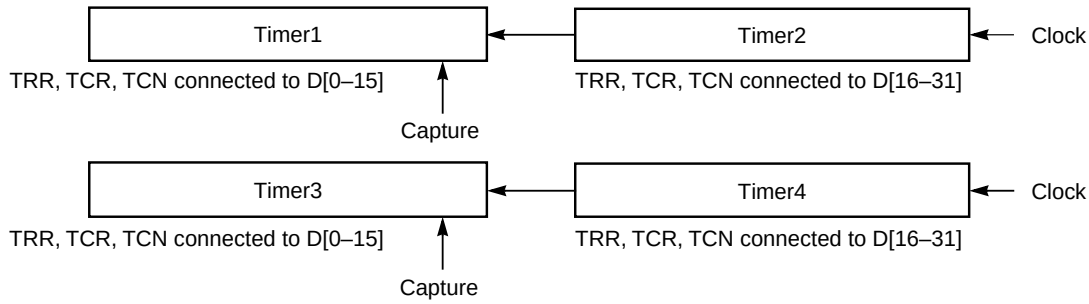


Figure 25-2. Timer Cascaded Mode Block Diagram

If TGCR[CAS] = 1, the two timers function as a 32-bit timer with a 32-bit TRR, TCR, and TCN. In this case, TMR1 and/or TMR3 are ignored, and the modes are defined using TMR2 and/or TMR4. The capture is controlled from TIN2 or TIN4 and the interrupts are generated from TER2 or TER4. In cascaded mode, the combined TRR, TCR, and TCN must be referenced with 32-bit bus cycles.

25.2.2 Timer Global Configuration Registers (TGCR1 and TGCR2)

The timer global configuration registers (TGCR1 and TGCR2), shown in Figure 25-3 and Figure 25-4, contain configuration parameters used by the timers. These registers allow simultaneous starting and stopping of a pair of timers (1 and 2 or 3 and 4) if one bus cycle is used.

	0	1	2	3	4	5	6	7
Field	CAS2	—	STP2	RST2	GM1	—	STP1	RST1
Reset	0000_0000							
R/W	R/W							
Addr	0x9_0D80							

Figure 25-3. Timer Global Configuration Register 1 (TGCR1)

Table 25-1 describes TGCR1 fields.

Table 25-1. TGCR1 Field Descriptions

Bits	Name	Description
0	**CAS2**	Cascade timers. 0 Normal operation. 1 Timers 1 and 2 cascade to form a 32-bit timer.
1	—	Reserved, should be cleared.

Table 25-1. TGCR1 Field Descriptions (continued)

Bits	Name	Description
2	**STP 2**	Stop timer. 0 Normal operation. 1 Reduce power consumption of the timer. This bit stops all clocks to the timer, except the clock from the internal bus interface, which allows the user to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
3	**RST2**	Reset timer. 0 Reset the corresponding timer (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP bit is cleared.
4	**GM1**	Gate mode for $\overline{\text{TGATE1}}$. This bit is valid only if the gate function is enabled in TMR1 or TMR2. 0 Restart gate mode. $\overline{\text{TGATE1}}$ is used to enable/disable count. A falling $\overline{\text{TGATE1}}$ enables and restarts the count and a rising edge of $\overline{\text{TGATE1}}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE1}}$ does not restart the count value in TCN.
5	—	Reserved, should be cleared.
6	**STP1**	Stop timer. 0 Normal operation. 1 Reduce power consumption of the timer. This bit stops all clocks to the timer, except the clock from the internal bus interface, which allows the user to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
7	**RST1**	Reset timer. 0 Reset the corresponding timer (a software reset is identical to an external reset). 1 Enable the corresponding timer if STP = 0.

The TGCR2 register is shown in Figure 25-4.

	0	1	2	3	4	5	6	7
Field	CAS4	—	STP4	RST4	GM2	—	STP3	RST3
Reset	0000_0000							
R/W	R/W							
Addr	0x9_0D84							

Figure 25-4. Timer Global Configuration Register 2 (TGCR2)

Table 25-2 describes TGCR2 fields.

Table 25-2. TGCR2 Field Descriptions

Bit	Name	Description
0	**CAS4**	Cascade timers. 0 Normal operation. 1 Timers 3 and 4 cascades to form a 32-bit timer.
1	—	Reserved, should be cleared.

Table 25-2. TGCR2 Field Descriptions (continued)

Bit	Name	Description
2	**STP 4**	Stop timer 0 Normal operation. 1 Reduce power consumption of the timer. This bit stops all clocks to the timer, except the clock from the internal bus interface, which allows the user to read and write timer registers. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
3	**RST4**	Reset timer 0 Reset the corresponding timer (a software reset is identical to an external reset). 1 Enable the corresponding timer if the STP bit is cleared.
4	**GM2**	Gate mode for $\overline{\text{TGATE2}}$. This bit is valid only if the gate function is enabled in TMR3 or TMR4. 0 Restart gate mode. $\overline{\text{TGATE2}}$ is used to enable/disable the count. The falling edge of $\overline{\text{TGATE2}}$ enables and restarts the count and the rising edge of $\overline{\text{TGATE2}}$ disables the count. 1 Normal gate mode. This mode is the same as 0, except the falling edge of $\overline{\text{TGATE2}}$ does not restart the count value in TCN.
5	—	Reserved, should be cleared.
6	**STP3**	Stop timer 0 Normal operation. 1 Reduce power consumption of the timer. This bit stops all clocks to the timer, however it is possible to read the values while the clock is stopped. The clocks to the timer remain stopped until the user clears this bit or a hardware reset occurs.
7	**RST3**	Reset timer 0 Reset the corresponding timer (a software reset is identical to an external reset). 1 Enable the corresponding timer if STP = 0.

25.2.3 Timer Mode Registers (TMR1–TMR4)

The four timer mode registers (TMR1–TMR4) are shown in Figure 25-5.

Erratic behavior may occur if TGCR1 and TGCR2 are not initialized before the TMRs. Only TGCR[RST] can be modified at any time.

	0 7	8 9	10	11	12	13 14	15
Field	PS	CE	OM	ORI	FRR	ICLK	GE
Reset	0000_0000_0000_0000						
R/W	R/W						
Addr	0x9_0D90 (TMR1); 0x9_0D92 (TMR2); 0x9_0DA0 (TMR3); 0x9_0DA2 (TMR4)						

Figure 25-5. Timer Mode Registers (TMR1–TMR4)

Table 25-3 describes TMR1–TMR4 register fields.

Table 25-3. TMR1–TMR4 Field Descriptions

Bits	Name	Description
0–7	**PS**	Prescaler value. The prescaler is programmed to divide the clock input by values from 1 to 256. The value 00000000 divides the clock by 1 and 11111111 divides the clock by 256.
8–9	**CE**	Capture edge and enable interrupt. 00 Disable interrupt on capture event; capture function is disabled. 01 Capture on rising TIN*x* edge only and enable interrupt on capture event. 10 Capture on falling TIN*x* edge only and enable interrupt on capture event. 11 Capture on any TIN*x* edge and enable interrupt on capture event.
10	**OM**	Output mode 0 Active-low pulse on $\overline{\text{TOUT}x}$ for one timer input clock cycle as defined by the ICLK bits. Thus, $\overline{\text{TOUT}x}$ may be low for one bus clock period, one bus clock/16 period, or one TIN*x* clock cycle period. $\overline{\text{TOUT}x}$ changes occur on the rising edge of the system clock. 1 Toggle $\overline{\text{TOUT}x}$. $\overline{\text{TOUT}x}$ changes occur on the rising edge of the system clock.
11	**ORI**	Output reference interrupt enable. 0 Disable interrupt for reference reached (does not affect interrupt on capture function). 1 Enable interrupt upon reaching the reference value.
12	**FRR**	Free run/restart. 0 Free run. The timer count continues to increment after the reference value is reached. 1 Restart. The timer count is reset immediately after the reference value is reached.
13–14	**ICLK**	Input clock source for the timer. 00 Internally cascaded input. For TMR1, the timer 1 input is the output of timer 2. For TMR3, the timer 3 input is the output of timer 4. For TMR2 and TMR4, this selection means no input clock is provided to the timer. 01 Internal bus clock. 10 Internal bus clock divided by 16. 11 Corresponding TIN*x*: TIN1, TIN2, TIN3, or TIN4 (falling edge).
15	**GE**	Gate enable. 0 $\overline{\text{TGATE}x}$ is ignored. 1 $\overline{\text{TGATE}x}$ is used to control the timer.

25.2.4 Timer Reference Registers (TRR1–TRR4)

Each timer reference register (TRR1–TRR4), shown in Figure 25-6, contains the timeout's reference value. The reference value is not reached until TCN*x* increments to equal the timeout reference value.

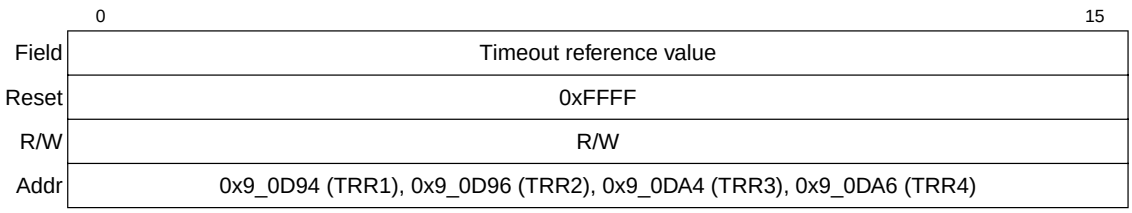


Figure 25-6. Timer Reference Registers (TRR1–TRR4)

25.2.5 Timer Capture Registers (TCR1–TCR4)

Each timer capture register (TCR1–TCR4), shown in Figure 25-7, is used to latch the value of the counter according to TMR*x*[CE].

	0	15
Field	Latched counter value	
Reset	0x0000	
R/W	R/W	
Addr	0x9_0D98 (TCR1), 0x9_0D9A (TCR2), 0x9_0DA8 (TCR3), 0x9_0DAA (TCR4)	

Figure 25-7. Timer Capture Registers (TCR1–TCR4)

25.2.6 Timer Counters (TCN1–TCN4)

Each timer counter register (TCN1–TCN4), shown in Figure 25-8, is an up-counter. A read cycle to TCN*x* yields the current value of the timer but does not affect the counting operation. A write cycle to TCN*x* sets the register to the written value, thus causing its corresponding prescaler, TMR*x*[PS], to be reset.

	0	15
Field	Up counter	
Reset	0x0000	
R/W	R/W	
Addr	0x9_0D9C (TCN1), 0x9_0D9E (TCN2), 0x9_0DAC (TCN3), 0x9_0DAE (TCN4)	

Figure 25-8. Timer Counter Registers (TCN1–TCN4)

Note that the counter registers may not be updated correctly if a write is made while the timer is not running. Use TRR*x* to define the preferred count value.

25.2.7 Timer Event Registers (TER1–TER4)

Each timer event register (TER*x*), shown in Figure 25-9, reports events recognized by the timers. When an output reference event is recognized, the timer sets TER*x*[REF] regardless of the corresponding TMR*x*[ORI]. The capture event is set only if it is enabled by TMR*x*[CE]. TER1–TER4 can be read at any time.

Writing ones clears event bits; writing zeros has no effect. Both event bits must be cleared before the timer negates the interrupt.

	0 … 13	14	15
Field	—	REF	CAP
Reset	0x0000		
Addr	0x9_0DB0 (TER1); 0x9_0DB2 (TER2); 0x9_0DB4 (TER3); 0x9_0DB6 (TER4)		

Figure 25-9. Timer Event Registers (TER1–TER4)

Table 25-4 describes TER fields.

Table 25-4. TER Field Descriptions

Bits	Name	Description
0–13	–	Reserved, should be cleared.
14	REF	Output reference event. The counter has reached the TRR value. TMR[ORI] is used to enable the interrupt request caused by this event.
15	CAP	Capture event. The counter value has been latched into the TCR. TMR[CE] is used to enable generation of this event.

Chapter 26
SDMA Channels

The MPC8560 has two physical serial DMA (SDMA) channels. The CP implements two dedicated virtual SDMA channels for each FCC, MCC, SCC, SPI, and I²C—one for each transmitter and receiver.

Table 26-1 shows data flow paths. Data from the peripheral controllers can be routed to external DDR SDRAM using the DDR bus (path 1) or the local bus (path 2).

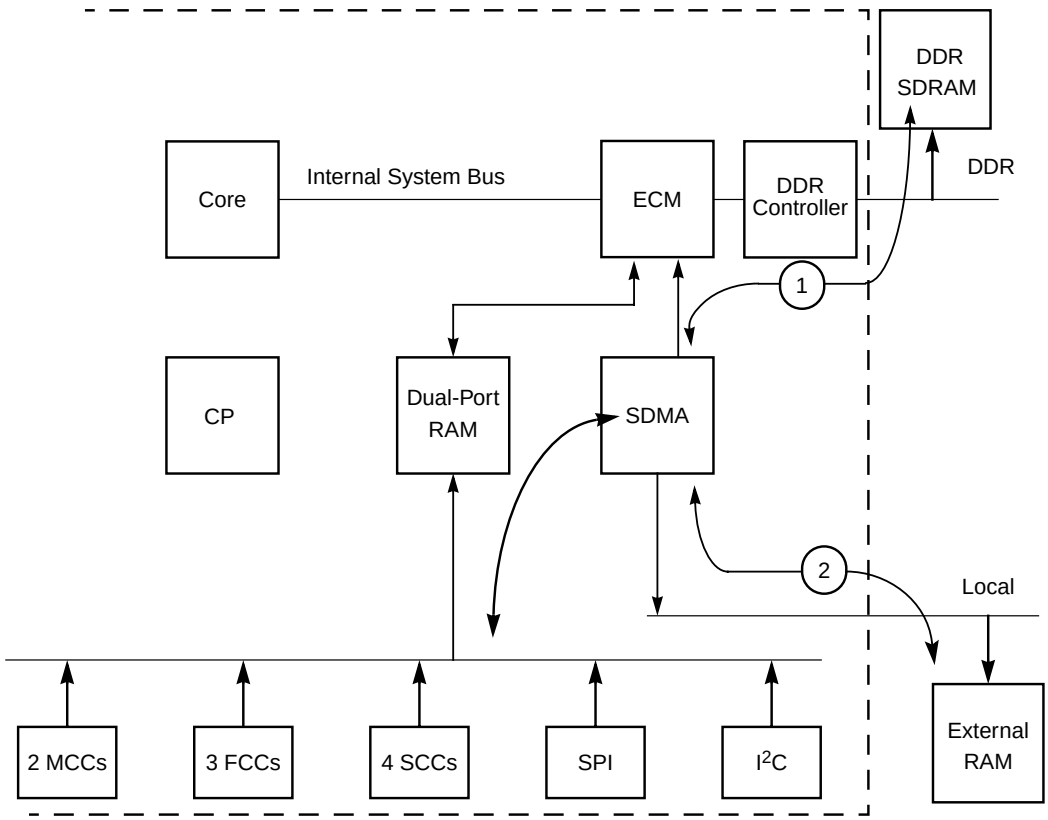


Figure 26-1. SDMA Data Paths

On a path 1 access, the SDMA channel must acquire the external DDR system bus. On a path 2 access, the local bus is acquired and the access is not seen on the external DDR system bus. Thus, the local bus transfer occurs at the same time as other operations on the external DDR system bus.

The SDMA channel can be assigned a big-endian (Motorola) or little-endian format for accessing buffer data. These features are programmed in the receive and transmit registers associated with the FCCs, MCCs, SCCs, SPI, and I²C.

If a system or local bus error occurs on a CP-related access by the SDMA, the CP generates a unique interrupt in the SDMA status register (SDSR). The interrupt service routine then reads the appropriate DMA transfer error address register (PDTEA for the system bus or LDTEA for the local bus) to determine the address the bus error occurred on. The channel that caused the bus error is determined by reading the channel number from PDTEM or LDTEM. If an SDMA bus error occurs on a CP-related transaction, all CPM activity stops and the entire CPM must be reset in the CP command register (CPCR). See Section 26.1, “SDMA Registers.”

26.1 SDMA Registers

The only user-accessible registers associated with the SDMA are the SDMA address registers, read-only register used for diagnostics in case of an SDMA bus error, the SDMA status register and the SDMA mask register.

26.1.1 SDMA Status Register (SDSR)

The SDMA status register (SDSR) reports bus error events recognized by the SDMA controller for all 22 SDMA channels. On recognition of a bus error on the local or system buses, the SDMA sets its corresponding SDSR bit. The SDSR is a memory-mapped register that can be read at any time. Bits are cleared by writing ones to them; writing zeros has no effect.

	0 – 5	6	7
Field	—	SBER_L	SBER_P
Reset	0000_0000		
Addr	0x9_1018		

Figure 26-2. SDMA Status Register (SDSR)

Table 26-1 describes SDSR fields.

Table 26-1. SDSR Field Descriptions

Bits	Name	Description
0–5	—	Reserved, should be cleared.
6	SBER_L	SDMA channel local bus error. Indicates that the SDMA channel on the local bus had terminated with an error during a read or write transaction. This bit is cleared writing a 1; writing a zero has no effect. The SDMA transfer error address can be read from LDTEA, and the channel number from LDTEM. Assertion of SBER_L causes the value inside LDTEA to stop updating. See Section 18.2.3, “SDMA Transfer Error Address Registers (PDTEA and LDTEA).
7	SBER_P	SDMA channel system bus error. Indicates that the SDMA channel on the system bus had terminated with an error during a read or write transaction. This bit is cleared writing a 1; writing a zero has no effect. The SDMA transfer error address is read from PDTEA. The channel number is read from PDTEM.

26.1.2 SDMA Mask Register (SDMR)

The SDMA mask register (SDMR) is an 8-bit read/write register with the same bit format as the SDMA status register. If an SDMR bit is 1, the corresponding interrupt in SDSR is enabled. If the bit is zero, the corresponding interrupt in the status register is masked. SDMR is cleared at reset. SDMR can be accessed at 0x9_101C.

26.1.3 Address Error Registers (SMAER and LMAER)

SMAER and LMAER indicate the address where the error occurs at read or write transaction. These registers hold the system or local address of the error transaction, respectively. Both registers can be cleared by hard reset. Note that soft reset has no effect on these registers. These registers are updated at the first occurrence of the error detection; the next update depends on whether the event register is cleared. Table 26-2 describes the parameters of the SMAER and LMAER registers.

Table 26-2. SMAER and LMAER Parameters

Parameters	Description
Bits	0–31
Filed	SMAER/LMAER
Reset	0000_0000_0000_0000_0000_0000_0000_0000
R/W	R
Address	0x10050 (SMAER); 0x10060 (LMAER)

26.1.4 Status Error Registers (SMSER and LMSER)

SMSER and LMSER indicate the status of the data error that occurs at read or write transaction. These registers hold the system or local status of the error transaction, respectively.

Both registers can be cleared by hard reset. Note that soft reset has no effect on these registers. These registers are updated at the first occurrence of the error detection; the next update depends on whether the event register is cleared. Table 26-3 describes the parameters of the SMSER and LMSER registers.

Table 26-3. SMSER and LMSER Parameters

Parameters	Description
Bits	0–31
Filed	SMSER/LMSER
Reset	0000_0000_0000_0000_0000_0000_0000_0000
R/W	R
Address	0x10054 (SMAER); 0x10064 (LMAER)

26.1.5 Event Registers (SMEVR and LMEVR)

SMEVR and LMEVR indicate the validity of the error that is written to SMSER/LMSER and SMAER/LMAER. These registers are updated when the first data error is detected after the read or write transaction. SMEVR and LMEVR can be cleared by writing one to bit zero. Note that writing zero to this bit has no effect. MSNUM is the serial number that causes the data error. Table 26-4 describes the parameters of the registers.

Table 26-4. SMEVR and LMEVR Parameters

Bits	0	1	2–7	8–31
Field	DATAERR	Reserved	MSNUM	Reserved
Reset	0	—	00_000	—
R/W	R/W	—	R	—
Address	0x10058 (SMEVR); 0x10068(LMEVR)			

Bit fields of these two registers are updated only when a read error occurs. Both registers can be cleared by hard reset. Note that soft reset has no effect on these registers. Table 26-5 describes the fields of SMEVR and LMEVR registers.

Table 26-5. SMEVR and LMEVR Field Descriptions

Bit(s)	Name	Description
0	DATAERR	Indicates data error. This bit is cleared by writing one to it. Writing zero to this bit has no effect. 0 No error occured 1 Error occurs during reading/writing
1	—	Reserved

Table 26-5. SMEVR and LMEVR Field Descriptions

Bit(s)	Name	Description
2–7	MSNUM	Serial number that causes the data error. It is updated only when a read error occurs.
8–31	—	Reserved

Chapter 27
Serial Communications Controllers (SCCs)

The MPC8560 has four serial communications controllers (SCCs), which can be configured independently to implement different protocols for bridging functions, routers, and gateways, and to interface with a wide variety of standard WANs, LANs, and proprietary networks. An SCC has many physical interface options such as interfacing to TDM buses, ISDN buses, and standard modem interfaces.

The SCCs are independent from the physical interface, but SCC logic formats and manipulates data from the physical interface. Furthermore, the choice of protocol is independent from the choice of interface. An SCC is described in terms of the protocol it runs. When an SCC is programmed to a certain protocol or mode, it implements functionality that corresponds to parts of the protocol's link layer (layer 2 of the OSI reference model). Many SCC functions are common to protocols of the following controllers:

- UART, described in Chapter 28, "SCC UART Mode."
- HDLC and HDLC bus, described in Chapter 29, "SCC HDLC Mode."
- AppleTalk/LocalTalk, described in Chapter 32, "SCC AppleTalk Mode."
- BISYNC, described in Chapter 30, "SCC BISYNC Mode."
- Transparent, described in Chapter 31, "SCC Transparent Mode."

Although the selected protocol usually applies both to the SCC transmitter and receiver, one half of an SCC can run transparent operations while the other runs a standard protocol.

Each Rx and Tx internal clock can be programmed with either an external or internal source. Internal clocks originate from one of eight baud rate generators (BRGs) or an external clock pin; see Section 23.3, "NMSI Configuration," for each SCC's available clock sources. These clocks can be as fast as a 1:4 ratio of the system clock. (For example, an SCC internal clock can run at 12.5 MHz in a 50-MHz system.) However, an SCC's ability to support a sustained bit stream depends on the protocol as well as other factors.

Associated with each SCC is a digital phase-locked loop (DPLL) for external clock recovery, which supports NRZ, NRZI, FM0, FM1, Manchester, and Differential Manchester. If the clock recovery function is not required (that is, synchronous

communication), then the DPLL can be disabled, in which case only NRZ and NRZI are supported.

An SCC can be connected to its own set of pins on the MPC8560. This configuration is called the non-multiplexed serial interface (NMSI) and is described in Chapter 22, “Serial Interface with Time-Slot Assigner.” Using NMSI, an SCC can support standard modem interface signals, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$. If required, software and additional parallel I/O lines can be used to support additional handshake signals. Figure 27-1 shows the SCC block diagram.

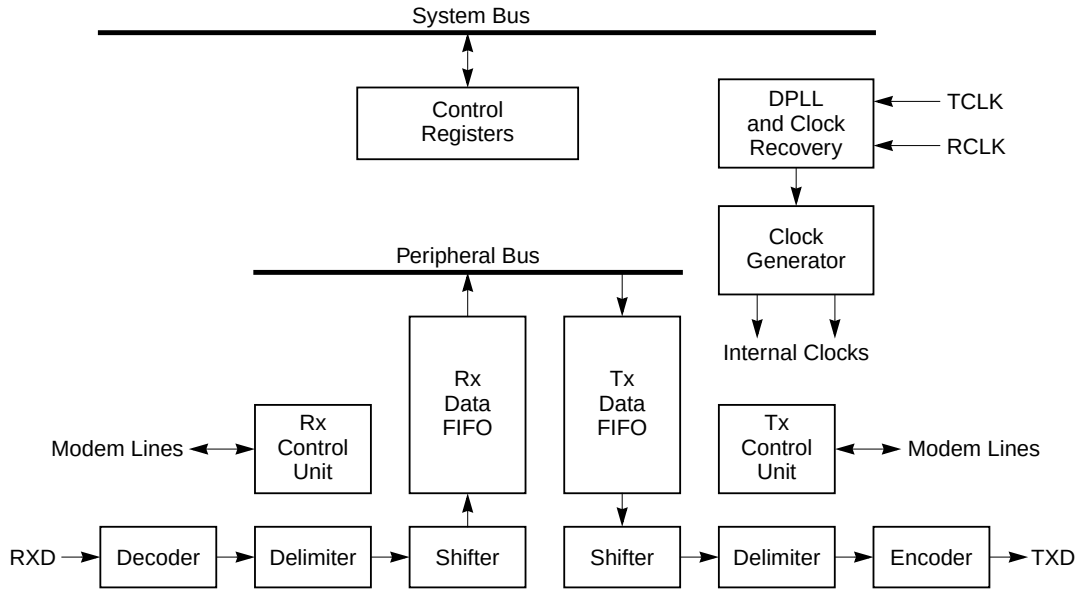


Figure 27-1. SCC Block Diagram

27.1 Features

The following is a list of the main SCC features. (Performance figures assume a 25-MHz system clock.)

- Implements HDLC/SDLC, HDLC bus, synchronous start/stop, asynchronous start/stop (UART), AppleTalk/LocalTalk, and totally transparent protocols
- Additional protocols supported through Motorola-supplied RAM microcodes: Profibus and Signaling System#7 (SS7)
- DPLL circuitry for clock recovery with NRZ, NRZI, FM0, FM1, Manchester, and Differential Manchester (also known as Differential Bi-phase-L)
- Clocks can be derived from a baud rate generator, an external pin, or DPLL
- Supports automatic control of the $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ modem signals

- Multi-buffer data structure for receive and send (the number of buffer descriptors (BDs) is limited only by the size of the internal dual-port RAM—8 bytes per BD)
- Deep FIFOs (SCC transmit and receive FIFOs are 32 bytes each.)
- Transmit-on-demand feature decreases time to frame transmission (transmit latency)
- Low FIFO latency option for send and receive in character-oriented and totally transparent protocols
- Frame preamble options
- Full-duplex operation
- Fully transparent option for one half of an SCC (Rx/Tx) while another protocol executes on the other half (Tx/Rx)
- Echo and local loopback modes for testing

27.2 General SCC Mode Registers (GSMR1–GSMR4)

Each SCC contains a general SCC mode register (GSMR) that defines options common to most of the protocols. GSMR_L contains the low-order 32 bits; GSMR_H, shown in Figure 27-2, contains the high-order 32 bits. Some GSMR operations are described in later sections.

	0	14	15
Field	—		
Reset	0000_0000_0000_0000		
R/W	R/W		
Addr	0x9_1A04 (GSMR1); 0x9_1A24 (GSMR2); 0x9_1A44 (GSMR3); 0x9_1A64 (GSMR4)		

	16 17	18	19	20	21	22	23	24	25	26	27	28 29	30	31
Field	TCRC	REVD	TRX	TTX	CDP	CTSP	CDS	CTSS	TFL	RFW	TXSY	SYNL	RTSM	RSYN
Reset	0000_0000_0000_0000													
R/W	R/W													
Addr	0x9_1A06 (GSMR1); 0x9_1A26 (GSMR2); 0x9_1A46 (GSMR3); 0x9_1A66 (GSMR4)													

Figure 27-2. GSMR_H—General SCC Mode Register (High Order)

Table 27-1 describes GSMR_H fields.

Table 27-1. GSMR_H Field Descriptions

Bit	Name	Description
0–15	—	Reserved, should be cleared.
16–17	TCRC	Transparent CRC (valid for totally transparent channel only). Selects the frame checking provided on transparent channels of the SCC (either the receiver, transmitter, or both, as defined by TTX and TRX). Although this configuration selects a frame check type, the decision to send the frame check is made in the TxBD. Thus, frame checks are not needed in transparent mode and frame check errors generated on the receiver can be ignored. 00 16-bit CCITT CRC (HDLC). (X16 + X12 + X5 + 1). 01 CRC16 (BISYNC). (X16 + X15 + X2 + 1). 10 32-bit CCITT CRC (Ethernet and HDLC). (X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 + 1). 11 Reserved.
18	REVD	Reverse data (valid for a totally transparent channel only) 0 Normal operation. 1 Reverses the bit order for totally transparent channels on this SCC (either the receiver, transmitter, or both) and sends the msb of each byte first. Section 30.11, “BISYNC Mode Register (PSMR),” describes reversing bit order in a BISYNC protocol.
19–20	TRX, TTX	Transparent receiver/transmitter. The receiver, transmitter, or both can use totally transparent operation, regardless of GSMR_L[MODE]. For example, to configure the transmitter as a UART and the receiver for totally transparent operations, set MODE = 0b0100 (UART), TTX = 0, and TRX = 1. 0 Normal operation. 1 The channel uses totally transparent mode, regardless of the protocol chosen in GSMR_L[MODE]. For full-duplex totally transparent operation, set both TTX and TRX. Note that an SCC cannot operate with half in Ethernet mode and half in transparent mode. That is, if MODE = 0b1100 (Ethernet), erratic operation occurs unless TTX = TRX.
21, 22	CDP, CTSP	$\overline{CD}$/$\overline{CTS}$ pulse. If this SCC is used in the TSA and is programmed in transparent mode, set CTSP and refer to Section 31.4.2, “Synchronization and the TSA,” for options on programming CDP. 0 Normal operation (envelope mode). $\overline{CD}$/$\overline{CTS}$ should envelope the frame. Negating $\overline{CD}$/$\overline{CTS}$ during reception causes a $\overline{CD}$/$\overline{CTS}$ lost error. 1 Pulse mode. Synchronization occurs when $\overline{CD}$/$\overline{CTS}$ is asserted; further $\overline{CD}$/$\overline{CTS}$ transitions do not affect reception.
23, 24	CDS, CTSS	$\overline{CD}$/$\overline{CTS}$ sampling. Determine synchronization characteristics of $\overline{CD}$ and $\overline{CTS}$. If the SCC is in transparent mode and is used in the TSA, CDS and CTSS must be set. Also, CDS and CTSS must be set for loopback testing in transparent mode. 0 $\overline{CD}$/$\overline{CTS}$ is assumed to be asynchronous with data. It is internally synchronized by the SCC, then data is received ($\overline{CD}$) or sent ($\overline{CTS}$) after several clock delays. 1 $\overline{CD}$/$\overline{CTS}$ is assumed to be synchronous with data, which speeds up operation. $\overline{CD}$ or $\overline{CTS}$ must transition while the Rx/Tx clock is low, at which time, the transfer begins. Useful for connecting MPC8560 in transparent mode since the $\overline{RTS}$ of one MPC8560 can connect directly to the $\overline{CD}$/$\overline{CTS}$ of another.
25	TFL	Transmit FIFO length. 0 Normal operation. The transmit FIFO is 32 bytes. 1 The Tx FIFO is 1 byte. This option is used with character-oriented protocols, such as UART, to ensure a minimum FIFO latency at the expense of performance.

Table 27-1. GSMR_H Field Descriptions (continued)

Bit	Name	Description
26	RFW	Rx FIFO width. 0 Receive FIFO is 32 bits wide for maximum performance; the Rx FIFO is 32 bytes. Data is not normally written to receive buffers until at least 32 bits are received. This configuration is required for HDLC-type protocols and is recommended for high-performance transparent protocols. 1 Low-latency operation. The receive FIFO is 8 bits wide, reducing the Rx FIFO to a quarter its normal size. This allows data to be written to the buffer as soon as a character is received, instead of waiting to receive 32 bits. This configuration must be chosen for character-oriented protocols, such as UART. It can also be used for low-performance, low-latency, transparent operation. However, it must not be used with HDLC, HDLC Bus, or AppleTalk because it causes erratic behavior.
27	TXSY	Transmitter synchronized to the receiver. Intended for X.21 applications where the transmitted data must begin an exact multiple of 8-bit periods after the received data arrives. 0 No synchronization between receiver and transmitter (default). 1 The transmit bit stream is synchronized to the receiver. Additionally, if RSYN = 1, transmission in totally transparent mode does not occur until the receiver synchronizes with the bit stream and $\overline{\text{CTS}}$ is asserted to the SCC. Assuming $\overline{\text{CTS}}$ is asserted, transmission begins 8 clocks after the receiver starts receiving data.
28–29	SYNL	Sync length (BISYNC and transparent mode only). See the data synchronization register (DSR) definition in Section 30.9, “Sending and Receiving the Synchronization Sequence,” (BISYNC) and Section 31.4.1.1, “In-Line Synchronization Pattern,” (transparent). 00 An external sync ($\overline{\text{CD}}$) is used instead of the sync pattern in the DSR. 01 4-bit sync. The receiver synchronizes on a 4-bit sync pattern stored in the DSR. This sync and additional syncs can be stripped by programming the SCC’s parameter RAM for character recognition. 10 8-bit sync. Should be chosen along with the BISYNC protocol to implement mono-sync. The receiver synchronizes on an 8-bit sync pattern in the DSR. 11 16-bit sync. Also called BISYNC. The receiver synchronizes on a 16-bit sync pattern stored in the DSR.
30	RTSM	$\overline{\text{RTS}}$ mode. Determines whether flags or idles are to be sent. Can be changed on-the-fly. 0 Send idles between frames as defined by the protocol and the TEND bit. $\overline{\text{RTS}}$ is negated between frames (default). 1 Send flags/syncs between frames according to the protocol. $\overline{\text{RTS}}$ is always asserted whenever the SCC is enabled.
31	RSYN	Receive synchronization timing (totally transparent mode only). 0 Normal operation. 1 If CDS = 1, $\overline{\text{CD}}$ should be asserted on the second bit of the Rx frame rather than on the first.

Figure 27-3 shows GSMR_L.

	0	1 2	3	4 5	6	7	8 10	11 12	13	14 15
Field	—	EDGE	TCI	TSNC	RINV	TINV	TPL	TPP	TEND	TDCR
Reset	0000_0000_0000_0000									
R/W	R/W									
Addr	0x9_1A00 (SCC1); 0x9_1A20 (SCC2); 0x9_1A40 (SCC3); 0x9_1A60 (SCC4)									

	16 17	18 20	21 23	24 25	26	27	28 31
Field	RDCR	RENC	TENC	DIAG	ENR	ENT	MODE
Reset	0000_0000_0000_0000						
R/W	R/W						
Addr	0x9_1A02 (SCC1); 0x9_1A22 (SCC2); 0x9_1A42 (SCC3); 0x9_1A62 (SCC4)						

Figure 27-3. GSMR_L—General SCC Mode Register (Low Order)

Table 27-2 describes GSMR_L fields.

Table 27-2. GSMR_L Field Descriptions

Bit	Name	Description
0	—	Reserved, should be cleared.
1–2	EDGE	Clock edge. Determines the clock edge the DPLL uses to adjust the receive sample point due to jitter in the received signal. Ignored in UART protocol or if the 1x clock mode is selected in RDCR. 00 Both the positive and negative edges are used for changing the sample point (default). 01 Positive edge. Only the positive edge of the received signal is used to change the sample point. 10 Negative edge. Only the negative edge of the received signal is used to change the sample point. 11 No adjustment is made to the sample point.
3	TCI	Transmit clock invert. 0 Normal operation. 1 Before it is used, the internal Tx clock (TCLK) is inverted by the SCC so it can clock data out one-half clock earlier (on the rising rather than the falling edge). In this case, the SCC offers a minimum and maximum rising clock edge-to-data specification. Data output by the SCC after the rising edge of an external Tx clock can be latched by the external receiver one clock cycle later on the next rising edge of the same Tx clock. In HDLC and Transparent mode, when TCI=0, data is sent on the falling edge; when TCI=1, on the rising edge. Recommended for HDLC and transparent operation when clock rates exceed 8 MHz to improve data setup time for the external transceiver.
4–5	TSNC	Transmit sense. Determines the amount of time the internal carrier sense signal stays active after the last transition on RXD, indicating that the line is free. For instance, AppleTalk can use TSNC to avoid a spurious CS-changed (SCCE[DCC]) interrupt that would otherwise occur during the frame sync sequence before the opening flags. If RDCR is configured to 1× clock mode, the delay is the greater of the two numbers listed. If RDCR is configured to 8×, 16×, or 32× mode, the delay is the smaller number. 00 Infinite. Carrier sense is always active (default). 01 14- or 6.5-bit times as determined by RDCR. 10 4- or 1.5-bit times as determined by RDCR (normally for AppleTalk). 11 3- or 1-bit times as determined by RDCR.

Table 27-2. GSMR_L Field Descriptions (continued)

Bit	Name	Description
6	RINV	DPLL Rx input invert data. Must be zero in HDLC bus mode or asynchronous UART mode. 0 Do not invert. 1 Invert data before sending it to the DPLL for reception. Used to produce FM1 from FM0 and NRZI space from NRZI mark or to invert the data stream in regular NRZ mode.
7	TINV	DPLL Tx input invert data. Must be zero in HDLC bus mode. 0 Do not invert. 1 Invert data before sending it to the DPLL for transmission. Used to produce FM1 from FM0 and NRZI space from NRZI mark and to invert the data stream in regular NRZ mode. In T1 applications, setting TINV and TEND creates a continuously inverted HDLC data stream.
8–10	TPL	Tx preamble length. Determines the length of the preamble configured by the TPP bits. 000 No preamble (default). 001 8 bits (1 byte). 010 16 bits (2 bytes). 011 32 bits (4 bytes). 100 48 bits (6 bytes). 10164 bits (8 bytes). 110 128 bits (16 bytes). 111 Reserved.
11–12	TPP	Tx preamble pattern. Determines what, if any, bit pattern should precede each Tx frame. The preamble pattern is sent before the first flag/sync of the frame. TPP is ignored in UART mode. The preamble length is programmed in TPL; the preamble pattern is typically sent to a receiving station that uses a DPLL for clock recovery. The receiving DPLL uses the regular preamble pattern to help it lock onto the received signal in a short, predictable time period. 00 All zeros. 01 Repetitive 10s. 10 Repetitive 01s. 11 All ones. Select this setting for LocalTalk operation.
13	TEND	Transmitter frame ending. Intended for NRZI transmitter encoding of the DPLL. TEND determines whether TXD should idle in a high state or in an encoded ones state (high or low). It can, however, be used with other encodings besides NRZI. 0 Default operation. TXD is encoded only when data is sent, including the preamble and opening and closing flags/syncs. When no data is available to send, the signal is driven high. 1 TXD is always encoded, even when idles are sent.
14–15	TDCR	Transmitter/receiver DPLL clock rate. If the DPLL is not used, choose 1× mode except in asynchronous UART mode where 8×, 16×, or 32× must be chosen. TDCR should match RDCR in most applications to allow the transmitter and receiver to use the same clock source. If an application uses the DPLL, the selection of TDCR/RDCR depends on the encoding/decoding. If communication is synchronous, select 1×. FM0/FM1, Manchester, and Differential Manchester require 8×, 16×, or 32×. If NRZ- or NRZI-encoded communication is asynchronous (that is, clock recovery required), select 8×, 16×, or 32×. The 8× option allows highest speed, whereas the 32× option provides the greatest resolution. 00 1× clock mode. Only NRZ or NRZI encodings/decodings are allowed. 01 8× clock mode. 10 16× clock mode. Normally chosen for UART and AppleTalk. 11 32× clock mode.
16–17	RDCR	

Table 27-2. GSMR_L Field Descriptions (continued)

Bit	Name	Description
18–20	RENC	Receiver decoding/transmitter encoding method. Select NRZ if DPLL is not used. RENC should equal TENC in most applications. 000 NRZ (default setting if DPLL is not used). Required for UART (synchronous or asynchronous). 001 NRZI Mark (set RINV/TINV also for NRZI space). 010 FM0 (set RINV/TINV also for FM1). 011 Reserved. 100 Manchester. 101 Reserved. 110 Differential Manchester (Differential Bi-phase-L). 111 Reserved.
21–23	TENC	
24–25	DIAG	Diagnostic mode. 00 Normal operation, $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ are under automatic control. Data is received through RXD and transmitted through TXD. The SCC uses modem signals to enable or disable transmission and reception. These timings are shown in Section 27.4.5, “Controlling SCC Timing with RTS, CTS, and CD.” 01 Local loopback mode. Transmitter output is connected internally to the receiver input, while the receiver and the transmitter operate normally. The value on RXD is ignored. If enabled, data appears on TXD, or the parallel I/O registers can be programmed to make TXD high. $\overline{\text{RTS}}$ can also be programmed to be disabled in the appropriate parallel I/O register. The transmitter and receiver must share the same clock source, but separate CLK*x* pins can be used if connected to the same external clock source. If external loopback is preferred, program DIAG for normal operation and externally connect TXD and RXD. Then, physically connect the control signals ($\overline{\text{RTS}}$ connected to $\overline{\text{CD}}$, and $\overline{\text{CTS}}$ grounded) or set the parallel I/O registers so $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are permanently asserted to the SCC by configuring the associated $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins as general-purpose I/O. 10 Automatic echo mode. The transmitter automatically resends received data bit-by-bit using the Rx clock provided. The receiver operates normally and receives data if $\overline{\text{CD}}$ is asserted. $\overline{\text{CTS}}$ is ignored. 11 Loopback and echo mode. Loopback and echo operation occur simultaneously. $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are ignored. See the loopback bit description above for clocking requirements. For TDM operation, the diagnostic mode is selected by SI*x*MR[SDM*x*]; see Section 22.5.2, “SI Mode Registers (SIxMR).”
26	ENR	Enable receive. Enables the receiver hardware state machine for this SCC. 0 The receiver is disabled and data in the Rx FIFO is lost. If ENR is cleared during reception, the receiver aborts the current character. 1 The receiver is enabled. ENR can be set or cleared, regardless of whether serial clocks are present. Section 27.4.8, “Reconfiguring the SCCs,” describes how to disable/enable an SCC. Note that other tools, including the ENTER HUNT MODE and CLOSE RXBD commands and the E bit of the Rx BD, data provide the capability to control the receiver.

Table 27-2. GSMR_L Field Descriptions (continued)

Bit	Name	Description
27	ENT	Enable transmit. Enables the transmitter hardware state machine for this SCC. 0 The transmitter is disabled. If ENT is cleared during transmission, the current character is aborted and TXD returns to the idle state. Data already in the Tx shift register is not sent. 1 The transmitter is enabled. ENT can be set or cleared, regardless of whether serial clocks are present. Section 27.4.8, "Reconfiguring the SCCs," describes how to disable/enable an SCC. Note that other tools, such as the STOP TRANSMIT, GRACEFUL STOP TRANSMIT, and RESTART TRANSMIT commands, the freeze option and $\overline{\text{CTS}}$ flow control option in UART mode, and the R bit of the TxBD, also provide the capability to control the transmitter.
28–31	MODE	Channel protocol mode. See also GSMR_H[TTX, TRX]. 0000 HDLC 0001 Reserved 0010 AppleTalk/LocalTalk 0011 SS7—reserved for RAM microcode 0100 UART 0101 Profibus—reserved for RAM microcode 0110 Reserved 0111 Reserved 1000 BISYNC 1001 Reserved 101x Reserved 1100 Reserved 11xx Reserved

27.2.1 Protocol-Specific Mode Register (PSMR)

The protocol implemented by an SCC is selected by its GSMR_L[MODE]. Each SCC has an additional protocol-specific mode register (PSMR) that configures it specifically for the chosen protocol. The PSMR fields are described in the specific chapters that describe each protocol. PSMRs are cleared at reset. PSMRs reside at the following addresses: 0x9_1A08 (PSMR1), 0x9_1A28 (PSMR2), 0x9_1A48 (PSMR3), and 0x9_1A68 (PSMR4).

27.2.2 Data Synchronization Register (DSR)

Each SCC has a data synchronization register (DSR) that specifies the pattern used for frame synchronization. The programmed value for DSR depends on the following protocols:

- UART—DSR is used to configure fractional stop bit transmission.
- BISYNC and transparent—DSR should be programmed with the sync pattern.
- HDLC—At reset, DSR defaults to 0x7E7E (two HDLC flags), so it does not need to be written.

Figure 27-4 shows the sync fields.

	0 – 7	8 – 15
Field	SYN2	SYN1
Reset	0111_1110	0111_1110
R/W	R/W	
Addr	0x9_1A0E (DSR1); 0x9_1A2E (DSR2); 0x9_1A4E (DSR3); 0x9_1A6E (DSR4)	

Figure 27-4. Data Synchronization Register (DSR)

27.2.3 Transmit-on-Demand Register (TODR)

In normal operation, if no frame is being sent by an SCC, the CP periodically polls the R bit of the next TxBD to see if a new frame/buffer is requested. Depending on the SCC configuration, this polling occurs every 8–32 serial Tx clocks. The transmit-on-demand option, selected in the transmit-on-demand register (TODR) shown in Figure 27-5, shortens the latency of the Tx buffer/frame and is useful in LAN-type protocols where maximum inter-frame gap times are limited by the protocol specification.

	0	1 – 15
Field	TOD	—
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x9_1A0C (TODR1); 0x9_1A2C (TODR2); 0x9_1A4C (TODR3); 0x9_1A6C (TODR4)	

Figure 27-5. Transmit-on-Demand Register (TODR)

The CP can be configured to begin processing a new frame/buffer without waiting the normal polling time by setting TODR[TOD] after TxBD[R] is set. Because this feature favors the specified TxBD, it may affect servicing of other SCC FIFOs. Therefore, transmitting on demand should only be used when a high-priority TxBD has been prepared and enough time has passed since the last g transmission. Table 27-3 describes TODR fields.

Table 27-3. TODR Field Descriptions

Bits	Name	Description
0	TOD	Transmit on demand. 0 Normal operation. 1 The CP 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 automatically after one serial clock, but transmitting on demand continues until an unprepared (R = 0) BD is reached. TOD does not need to be set again if new TxBDs are added to the BD table as long as older TxBDs are still being processed. New TxBDs are processed in order. The first bit of the frame is typically clocked out 5-6 bit times after TOD is set.
1–15	—	Reserved, should be cleared.

27.3 SCC Buffer Descriptors (BDs)

Data associated with each SCC channel is stored in buffers and each buffer is referenced by a buffer descriptor (BD) that can reside anywhere in dual-port RAM. The total number of 8-byte BDs is limited only by the size of the dual-port RAM (128 BDs/1 Kbyte). These BDs are shared among all serial controllers—SCCs, SPI, and I^2C. The user defines how the BDs are allocated among the controllers.

Each 64-bit BD has the following structure:

- The half word at offset + 0x0 contains status and control bits that control and report on the data transfer. These bits vary from protocol to protocol. The CPM updates the status bits after the buffer is sent or received.
- The half word at offset + 0x2 (data length) holds the number of bytes sent or received.
  - For an RxBD, this is the number of bytes the controller writes into the buffer. The CPM writes the length after received data is placed into the associated buffer and the buffer closed. In frame-based protocols (but not including SCC transparent operation), this field contains the total frame length, including CRC bytes. Also, if a received frame's length, including CRC, is an exact multiple of MRBLR, the last BD holds no actual data but does contain the total frame length.
  - For a TxBD, this is the number of bytes the controller should send from its buffer. Normally, this value should be greater than zero. The CPM never modifies this field.
- The word at offset + 0x4 (buffer pointer) points to the beginning of the buffer in memory (internal or external).
  - For an RxBD, the value must be a multiple of four. (word-aligned)
  - For a TxBD, this pointer can be even or odd.

Shown in Figure 27-6, the format of Tx and Rx BDs is the same in each SCC mode. Only the status and control bits differ for each protocol.

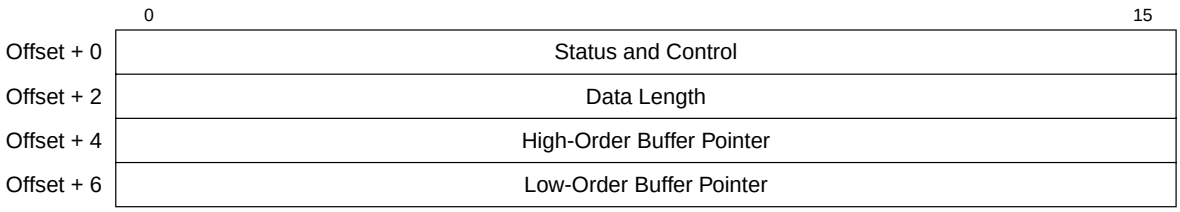


Figure 27-6. SCC Buffer Descriptors (BDs)

For frame-oriented protocols, a message can reside in as many buffers as necessary. Each buffer has a maximum length of 65,535 bytes. The CPM does not assume that all buffers of a single frame are currently linked to the BD table. The CPM does assume, however, that the unlinked buffers are provided by the core in time to be sent or received; otherwise, an

error condition is reported—an underrun error when sending and a busy error when receiving. Figure 27-7 shows the SCC BD table and buffer structure.

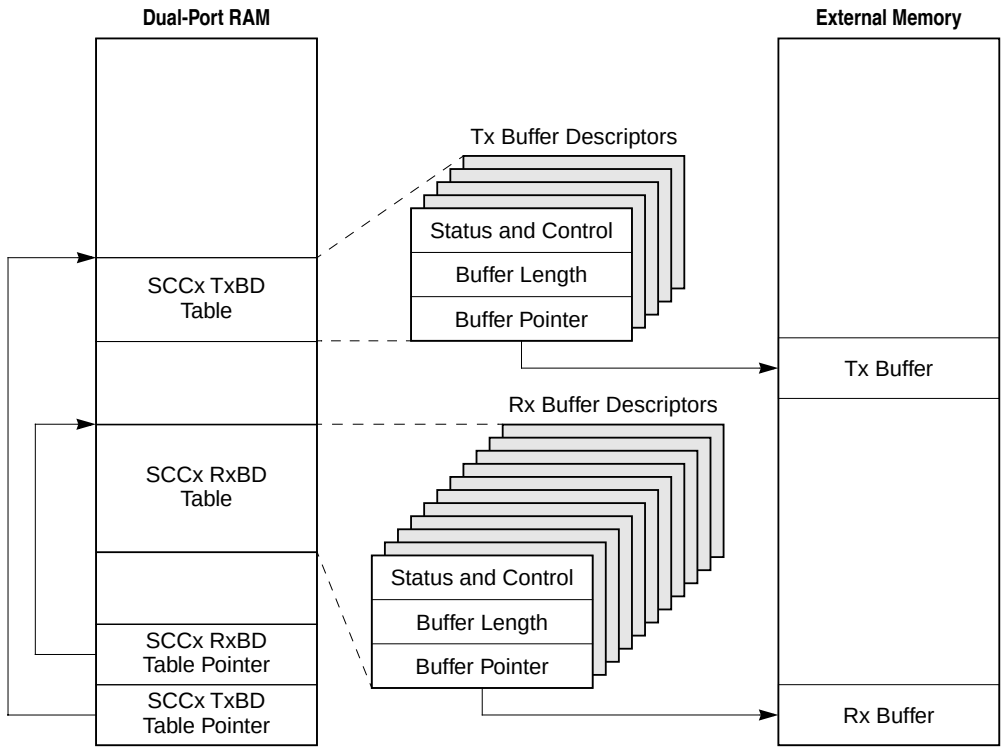


Figure 27-7. SCC BD and Buffer Memory Structure

In all protocols, BDs can point to buffers in the internal dual-port RAM. However, because dual-port RAM is used for descriptors, buffers are usually put in external RAM, especially if they are large.

The CPM processes TxBDs straightforwardly; when the transmit side of an SCC is enabled, the CPM starts with the first BD in that SCC TxBD table. Once the CPM detects that the R bit is set in the TxBD, it starts processing the buffer. The CPM detects that the BD is ready when it polls the R bit or when the user writes to the TODR. After data from the BD is put in the Tx FIFO, if necessary the CPM waits for the next descriptor's R bit to be set before proceeding. Thus, the CPM does no look-ahead descriptor processing and does not skip BDs that are not ready. When the CPM sees a BD's W bit (wrap) set, it returns to the start of the BD table after this last BD of the table is processed. The CPM clears R (not ready) after using a TxBD, which keeps it from being retransmitted before it is confirmed by the core. However, some protocols support a continuous mode (CM), for which R is not cleared (always ready).

The CPM uses RxBDs similarly. When data arrives, the CPM performs required processing on the data and moves resultant data to the buffer pointed to by the first BD; it continues until the buffer is full or an event, such as an error or end-of-frame detection, occurs. The

buffer is then closed; subsequent data uses the next BD. If E = 0, the current buffer is not empty and it reports a busy error. The CPM does not move from the current BD until E is set by the core (the buffer is empty). After using a descriptor, the CPM clears E (not empty) and does not reuse a BD until it has been processed by the core. However, in continuous mode (CM), E remains set. When the CPM discovers a descriptor's W bit set (indicating it is the last BD in the circular BD table), it returns to the beginning of the table when it is time to move to the next buffer.

27.4 SCC Parameter RAM

Each SCC parameter RAM area begins at the same offset from each SCC base area. Section 27.4.1, "SCC Base Addresses," describes the SCC's base addresses. The protocol-specific portions of the SCC parameter RAM are discussed in the specific protocol descriptions and the part that is common to all SCC protocols is shown in Table 27-4.

Some parameter RAM values must be initialized before the SCC can be enabled. Other values are initialized or written by the CPM. Once initialized, most parameter RAM values do not need to be accessed because most activity centers around the descriptors rather than the parameter RAM. However, if the parameter RAM is accessed, note the following:

- Parameter RAM can be read at any time.
- Tx parameter RAM can be written only when the transmitter is disabled—after a STOP TRANSMIT command and before a RESTART TRANSMIT command or after the buffer/frame finishes transmitting after a GRACEFUL STOP TRANSMIT command and before a RESTART TRANSMIT command.
- Rx parameter RAM can be written only when the receiver is disabled. Note the CLOSE RXBD command does not stop reception, but it does allow the user to extract data from a partially full Rx buffer.
- See Section 27.4.8, "Reconfiguring the SCCs."

Table 27-4 shows the parameter RAM map for all SCC protocols. Boldfaced entries must be initialized by the user.

Table 27-4. SCC Parameter RAM Map for All Protocols

Offset [1]	Name	Width	Description
0x00	**RBASE**	Hword	Rx/TxBD table base address—offset from the beginning of dual-port RAM. The BD tables can be placed in any unused portion of the dual-port RAM. The CPM starts BD processing at the top of the table. (The user defines the end of the BD table by setting the W bit in the last BD to be processed.) Initialize these entries before enabling the corresponding channel. Erratic operations occur if BD tables of active SCCs overlap. Values in RBASE and TBASE should be multiples of eight.
0x02	**TBASE**	Hword	
0x04	**RFCR**	Byte	Rx function code. See Section 27.4.2, "Function Code Registers (RFCR and TFCR)."
0x05	**TFCR**	Byte	Tx function code. See Section 27.4.2, "Function Code Registers (RFCR and TFCR)."

Table 27-4. SCC Parameter RAM Map for All Protocols (continued)

0x06	**MRBLR**	Hword	Maximum receive buffer length. Defines the maximum number of bytes the MPC8560 writes to a receive buffer before it goes to the next buffer. The MPC8560 can write fewer bytes than MRBLR if a condition such as an error or end-of-frame occurs. It never writes more bytes than the MRBLR value. Therefore, user-supplied buffers should be no smaller than MRBLR. MRBLR should be greater than zero for all modes. It should be a multiple of 4 for HDLC mode, and in totally transparent mode unless the Rx FIFO is 8-bits wide (GSMR_H[RFW] = 1). Note that although MRBLR is not intended to be changed while the SCC is operating, it can be changed dynamically in a single-cycle, 16-bit move (not two 8-bit cycles). Changing MRBLR has no immediate effect. To guarantee the exact Rx BD on which the change occurs, change MRBLR only while the receiver is disabled. Transmit buffer length is programmed in TxBD[Data Length] and is not affected by MRBLR.
0x08	RSTATE	Word	Rx internal state[3]
0x0C		Word	Rx internal buffer pointer [2]. The Rx and Tx internal buffer pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.
0x10	RBPTR	Hword	Current RxBD pointer. Points to the current BD being processed or to the next BD the receiver uses when it is idling. After reset or when the end of the BD table is reached, the CPM initializes RBPTR to the value in the RBASE. Although most applications do not need to write RBPTR, it can be modified when the receiver is disabled or when no Rx buffer is in use.
0x12		Hword	Rx internal byte count [2]. The Rx internal byte count is a down-count value initialized with MRBLR and decremented with each byte written by the supporting SDMA channel.
0x14		Word	Rx temp[3]
0x18	TSTATE	Word	Tx internal state[3]
0x1C		Word	Tx internal buffer pointer [2]. The Rx and Tx internal buffer pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.
0x20	TBPTR	Hword	Current TxBD pointer. Points to the current BD being processed or to the next BD the transmitter uses when it is idling. After reset or when the end of the BD table is reached, the CPM initializes TBPTR to the value in the TBASE. Although most applications do not need to write TBPTR, it can be modified when the transmitter is disabled or when no Tx buffer is in use (after a STOP TRANSMIT or GRACEFUL STOP TRANSMIT command is issued and the frame completes its transmission).
0x22		Hword	Tx internal byte count [2]. A down-count value initialized with TxBD[Data Length] and decremented with each byte read by the supporting SDMA channel.
0x24		Word	Tx temp[3]
0x28	RCRC	Word	Temp receive CRC [2]
0x2C	TCRC	Word	Temp transmit CRC [2]
0x30			Protocol-specific area. (The size of this area depends on the protocol chosen.)

[1] From SCC base. See Section 27.4.1, “SCC Base Addresses.”
[2] These parameters need not be accessed for normal operation but may be helpful for debugging.
[3] For CP use only

27.4.1 SCC Base Addresses

The CPM maintains a section of RAM called the parameter RAM, which contains many parameters for the operation of the FCCs, SCCs, SPI, and I²C. SCC base addresses are described in Table 27-5.

The exact definition of the parameter RAM is contained in each protocol subsection describing a device that uses a parameter RAM.

Table 27-5. Parameter RAM—SCC Base Addresses

Page	Address [1]	Peripheral	Size (Bytes)
1	0x8000	SCC1	256
2	0x8100	SCC2	256
3	0x8200	SCC3	256
4	0x8300	SCC4	256

[1] Offset from RAM_Base

27.4.2 Function Code Registers (RFCR and TFCR)

There are eight separate function code registers for the four SCC channels, four for Rx buffers (RFCR1–RFCR4) and four for Tx buffers (TFCR1–TFCR4). The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. Figure 27-8 shows the register format.

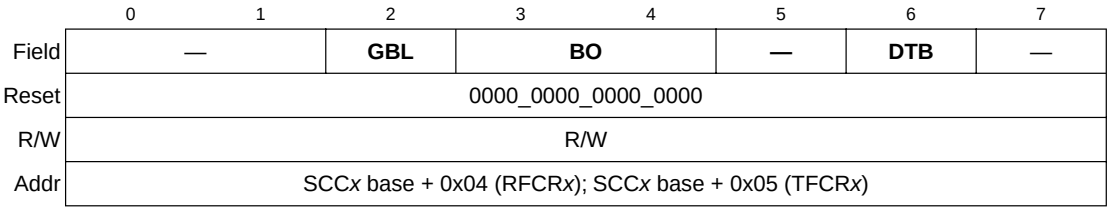


Figure 27-8. Function Code Registers (RFCR and TFCR)

Table 27-6 describes RFCR*x*/TFCR*x* fields.

Table 27-6. RFCR*x* /TFCR*x* Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	**GBL**	Global 0 Snooping disabled. 1 Snooping enabled.
3–4	**BO**	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame (HDLC and transparent) or at the beginning of the next BD. 0x Reserved 1x Big-endian
5	**—**	Reserved, should be cleared
6	**DTB**	Data bus indicator 0 Use system bus for SDMA operation 1 Use local bus for SDMA operation
7	—	Reserved, should be cleared.

27.4.3 Handling SCC Interrupts

To allow interrupt handling for SCC-specific events, event, mask, and status registers are provided within each SCC's internal memory map area; see Table 27-7. Because interrupt events are protocol-dependent, event descriptions are found in the specific protocol chapters.

Table 27-7. SCC*x* Event, Mask, and Status Registers

Register & IMMR Offset	Description
SCCE*x* 0x9_1A10 (SCCE1); 0x9_1A30 (SCCE2); 0x9_1A50 (SCCE3); 0x9_1A70 (SCCE4)	SCC event register. This 16-bit register reports events recognized by any of the SCCs. When an event is recognized, the SCC sets its corresponding bit in SCCE, regardless of the corresponding mask bit. When the corresponding event occurs, an interrupt is signaled to the SIVEC register. Bits are cleared by writing ones (writing zeros has no effect). SCCE is cleared at reset and can be read at any time.
SCCM*x* 0x9_1A14 (SCCM1); 0x9_1A34 (SCCM2); 0x9_1A54 (SCCM3); 0x9_1A74 (SCCM4)	SCC mask register. The 16-bit, read/write register allows interrupts to be enabled or disabled using the CPM for specific events in each SCC channel. An interrupt is generated only if SCC interrupts in this channel are enabled in the SIU interrupt mask register (SIMR). If an SCCM bit is zero, the CPM does not proceed with interrupt handling when that event occurs. The SCCM and SCCE bit positions are identical.
SCCS*x* 0x9_1A17 (SCCS1); 0x9_1A37 (SCCS2); 0x9_1A57 (SCCS3); 0x9_1A77 (SCCS4)	SCC status register. This 8-bit, read-only register allows monitoring of the real-time status of RXD.

Follow these steps to handle an SCC interrupt:

1. When an interrupt occurs, read SCCE to determine the interrupt sources and clear those SCCE bits (in most cases).
2. Process the TxBDs to reuse them if SCCE[TX] or SCCE[TXE] = 1. If the transmit speed is fast or the interrupt delay is long, the SCC may have sent more than one Tx buffer. Thus, it is important to check more than one TxBD during interrupt handling. A common practice is to process all TxBDs in the handler until one is found with its R bit set.
3. Extract data from the RxBD if SCCE[RX], SCCE[RXB], or SCCE[RXF] is set. As with transmit buffers, if the receive speed is fast or the interrupt delay is long, the SCC may have received more than one buffer and the handler should check more than one RxBD. A common practice is to process all RxBDs in the interrupt handler until one is found with RxBD[E] set.

- Execute the **rfi** instruction.

For additional information about interrupt handling refer to Chapter 21, “CPM Interrupt Controller.”

27.4.4 Initializing the SCCs

The SCCs require that a number of registers and parameters be configured after a power-on reset. Regardless of the protocol used, follow these steps to initialize SCCs:

1. Write the parallel I/O ports to configure and connect the I/O pins to the SCCs.
2. Configure the parallel I/O registers to enable $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ if these signals are required.
3. If the time-slot assigner (TSA) is used, the serial interface (SI*x*) must be configured. If the SCC is used in NMSI mode, CMXSCR must still be initialized.
4. Write all GSMR bits except ENT or ENR.
5. Write the PSMR.
6. Write the DSR.
7. Initialize the required values for this SCC’s parameter RAM.
8. Initialize the transmit/receive parameters via the CP command register (CPCR).
9. Clear out any current events in SCCE (optional).
10. Write ones to SCCM register to enable interrupts.
11. Set GSMR_L[ENT] and GSMR_L[ENR].

Descriptors can have their R or E bits set at any time. Notice that the CPCR does not need to be accessed after a hardware reset. An SCC should be disabled and reenabled after any dynamic change to its parallel I/O ports or serial channel physical interface configuration. A full reset can also be implemented using CPCR[RST].

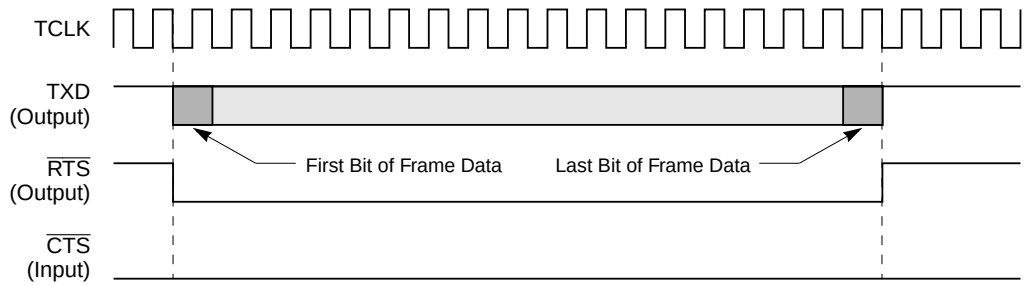
27.4.5 Controlling SCC Timing with $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$

When GSMR_L[DIAG] is programmed to normal operation, $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are controlled by the SCC. In the following subsections, it is assumed that GSMR_L[TCI] is zero, implying normal transmit clock operation.

27.4.5.1 Synchronous Protocols

$\overline{\text{RTS}}$ is asserted when the SCC data is loaded into the Tx FIFO and a falling Tx clock occurs. At this point, the SCC starts sending data once appropriate conditions occur on $\overline{\text{CTS}}$. In all cases, the first data bit is the start of the opening flag, sync pattern, or preamble.

Figure 27-9 shows that the delay between $\overline{\text{RTS}}$ and data is 0 bit times, regardless of GSMR_H[CTSS]. This operation assumes that $\overline{\text{CTS}}$ is already asserted to the SCC or that $\overline{\text{CTS}}$ is reprogrammed to be a parallel I/O line, in which case $\overline{\text{CTS}}$ to the SCC is always asserted. $\overline{\text{RTS}}$ is negated one clock after the last bit in the frame.



NOTE:
1. A frame includes opening and closing flags and syncs, if present in the protocol.

Figure 27-9. Output Delay from $\overline{\text{RTS}}$ Asserted for Synchronous Protocols

When $\overline{\text{RTS}}$ is asserted, if $\overline{\text{CTS}}$ is not already asserted, delays to the first data bit depend on when $\overline{\text{CTS}}$ is asserted. Figure 27-10 shows that the delay between $\overline{\text{CTS}}$ and the data can be approximately 0.5 to 1 bit times or no delay, depending on GSMR_H[CTSS].

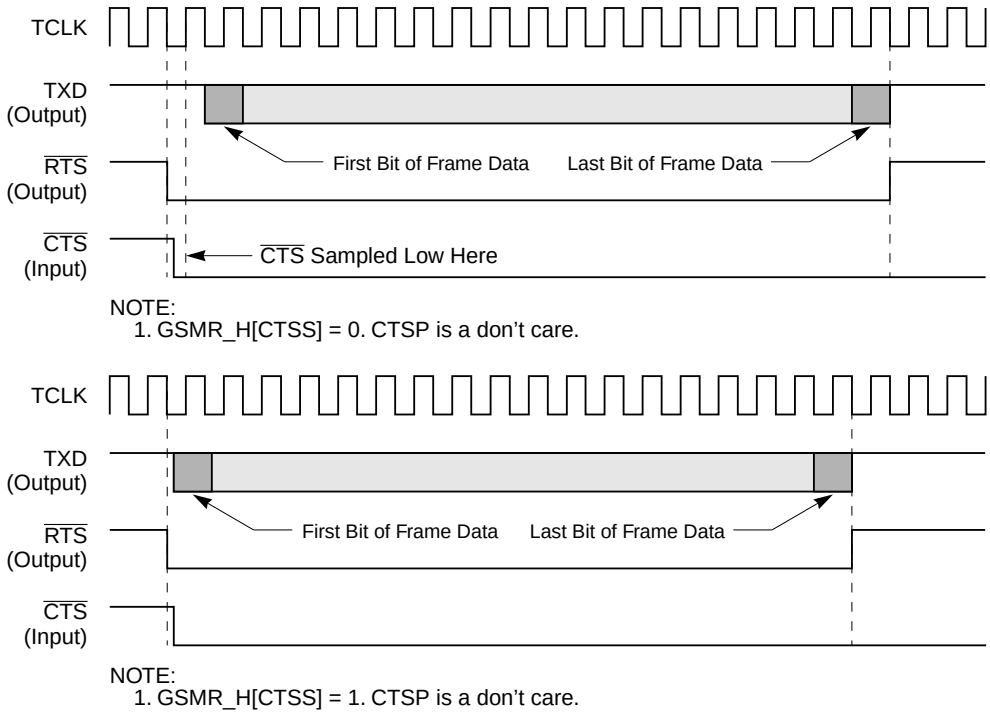


Figure 27-10. Output Delay from $\overline{\text{CTS}}$ Asserted for Synchronous Protocols

If $\overline{\text{CTS}}$ is programmed to envelope data, negating it during frame transmission causes a $\overline{\text{CTS}}$ lost error. Negating $\overline{\text{CTS}}$ forces $\overline{\text{RTS}}$ high and Tx data to become idle. If GSMR_H[CTSS] is zero, the SCC must sample $\overline{\text{CTS}}$ before a $\overline{\text{CTS}}$ lost is recognized; otherwise, the negation of $\overline{\text{CTS}}$ immediately causes the $\overline{\text{CTS}}$ lost condition. See Figure 27-11.

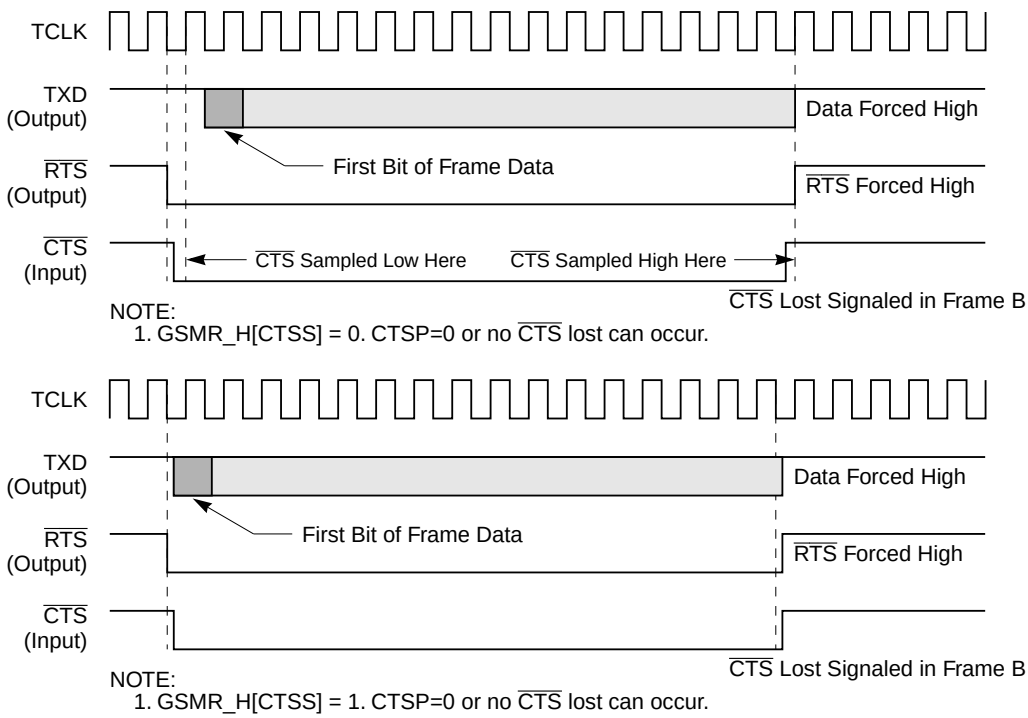


Figure 27-11. $\overline{\text{CTS}}$ Lost in Synchronous Protocols

Note that if GSMR_H[CTSS] = 1, $\overline{\text{CTS}}$ transitions must occur while the Tx clock is low.

Reception delays are determined by $\overline{\text{CD}}$ as shown in Figure 27-12. If GSMR_H[CDS] is zero, $\overline{\text{CD}}$ is sampled on the rising Rx clock edge before data is received. If GSMR_H[CDS] is 1, $\overline{\text{CD}}$ transitions cause data to be immediately gated into the receiver.

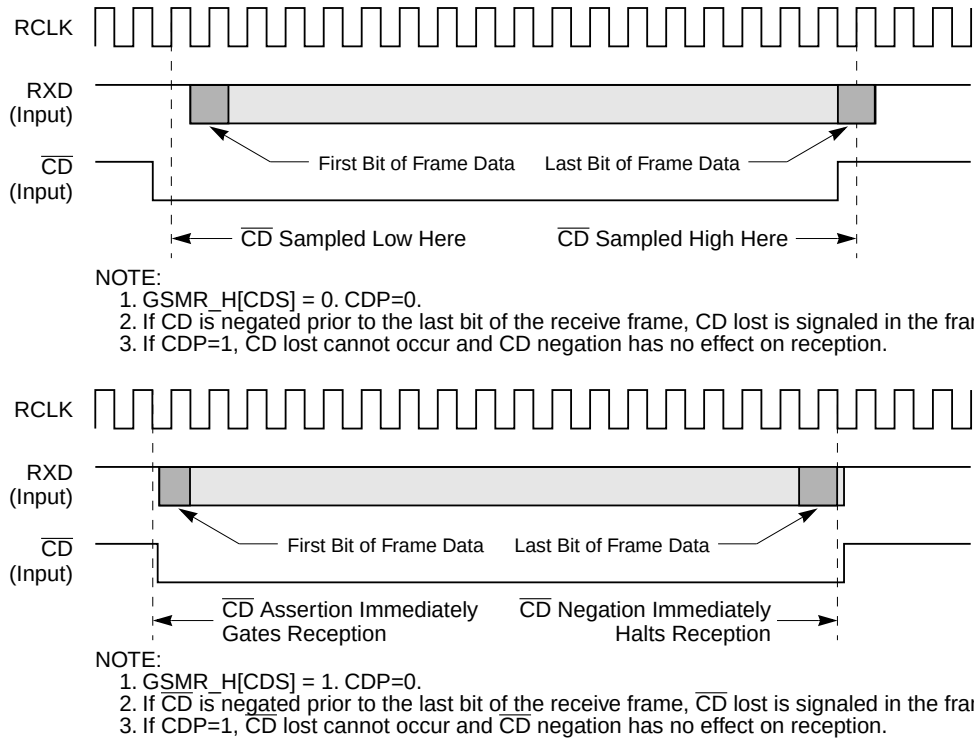


Figure 27-12. Using $\overline{CD}$ to Control Synchronous Protocol Reception

If $\overline{CD}$ is programmed to envelope the data, it must remain asserted during frame transmission or a $\overline{CD}$ lost error occurs. Negation of $\overline{CD}$ terminates reception. If GSMR_H[CDS] is zero, $\overline{CD}$ must be sampled by the SCC before a $\overline{CD}$ lost error is recognized; otherwise, the negation of $\overline{CD}$ immediately causes the $\overline{CD}$ lost condition.

If GSMR_H[CDS] is set, all $\overline{CD}$ transitions must occur while the Rx clock is low.

27.4.5.2 Asynchronous Protocols

In asynchronous protocols, $\overline{RTS}$ is asserted when SCC data is loaded into the Tx FIFO and a falling Tx clock occurs. $\overline{CD}$ and $\overline{CTS}$ can be used to control reception and transmission in the same manner as the synchronous protocols. The first bit sent in an asynchronous protocol is the start bit of the first character. In addition, the UART protocol has an option for $\overline{CTS}$ flow control as described in Chapter 28, “SCC UART Mode.”

- If $\overline{CTS}$ is already asserted when $\overline{RTS}$ is asserted, transmission begins in two additional bit times.
- If $\overline{CTS}$ is not already asserted when $\overline{RTS}$ is asserted and GSMR_H[CTSS] = 0, transmission begins in three additional bit times.
- If $\overline{CTS}$ is not already asserted when $\overline{RTS}$ is asserted and GSMR_H[CTSS] = 1, transmission begins in two additional bit times.

27.4.6 Digital Phase-Locked Loop (DPLL) Operation

Each SCC channel includes a digital phase-locked loop (DPLL) for recovering clock information from a received data stream. For applications that provide a direct clock source to the SCC, the DPLL can be bypassed by selecting 1x mode for GSMR_L[RDCR, TDCR]. If the DPLL is bypassed, only NRZ or NRZI encodings are available. The DPLL is optional for protocols. Figure 27-13 shows the DPLL receiver block; Figure 27-14 shows the transmitter block diagram.

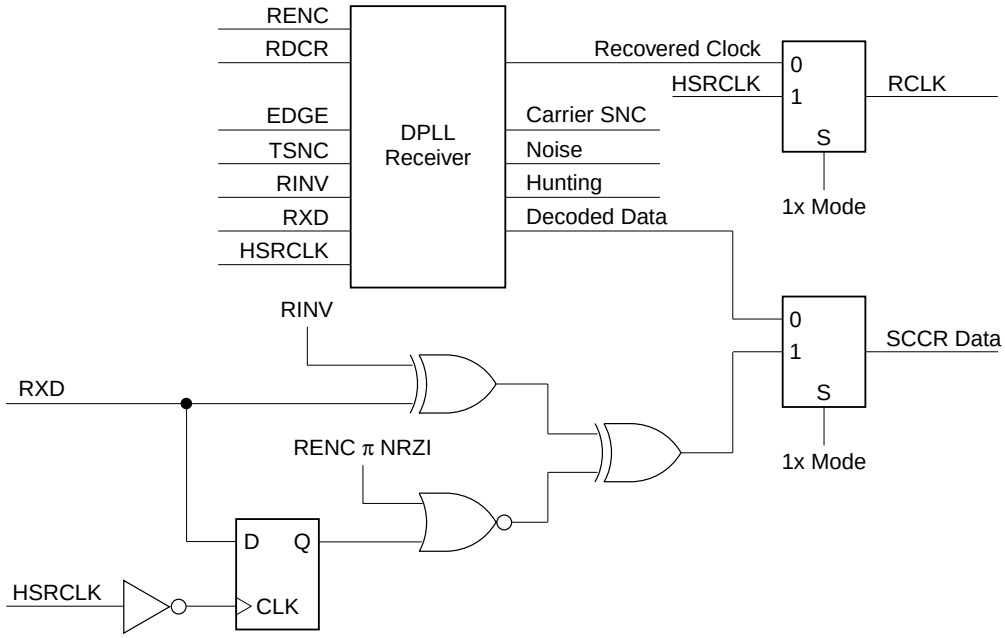


Figure 27-13. DPLL Receiver Block Diagram

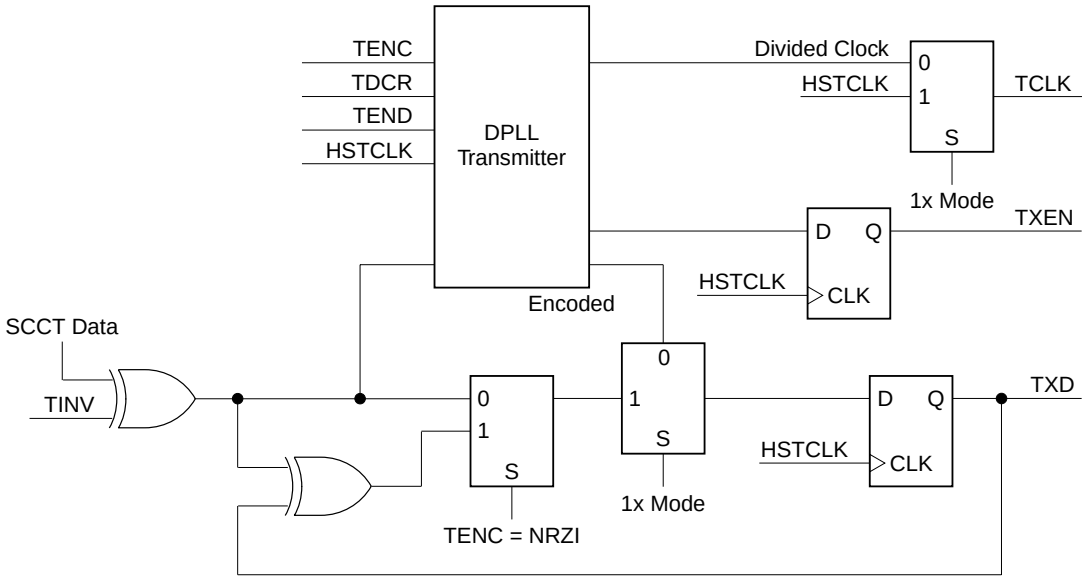


Figure 27-14. DPLL Transmitter Block Diagram

The DPLL can be driven by one of the baud rate generator outputs or an external clock, CLK*x*. In the block diagrams, this clock is labeled HSRCLK/HSTCLK. The HSRCLK/HSTCLK should be approximately 8x, 16x, or 32x the data rate, depending on the coding chosen. The DPLL uses this clock, along with the data stream, to construct a data clock that can be used as the SCC Rx and/or Tx clock. In all modes, the DPLL uses the input clock to determine the nominal bit time. If the DPLL is bypassed, HSRCLK/HSTCLK is used directly as RCLK/TCLK.

At the beginning of operation, the DPLL is in search mode, whereas the first transition resets the internal DPLL counter and begins DPLL operation. While the counter is counting, the DPLL watches the incoming data stream for transitions; when one is detected, the DPLL adjusts the count to produce an output clock that tracks incoming bits.

The DPLL has a carrier-sense signal that indicates when data transfers are on RXD. The carrier-sense signal asserts as soon as a transition is detected on RXD; it negates after the programmed number of clocks in GSMR_L[TSNC] when no transitions are detected.

To prevent itself from locking on the wrong edges and to provide fast synchronization, the DPLL should receive a preamble pattern before it receives the data. In some protocols, the preceding flags or syncs can function as a preamble; others use the patterns in Table 27-8. When transmission occurs, the SCC can generate preamble patterns, as programmed in GSMR_L[TPP, TPL].

Table 27-8. Preamble Requirements

Decoding Method	Preamble Pattern	Minimum Preamble Length Required
NRZI Mark	All zeros	8-bit
NRZI Space	All ones	8-bit
FM0	All ones	8-bit
FM1	All zeros	8-bit
Manchester	101010...10	8-bit
Differential Manchester	All ones	8-bit

The DPLL can also be used to invert the data stream of a transfer. This feature is available in all encodings, including standard NRZ format. Also, when the transmitter is idling, the DPLL can either force TXD high or continue encoding the data supplied to it.

The DPLL is used for UART encoding/decoding, which gives the option of selecting the divide ratio in the UART decoding process (8×, 16×, or 32×). Typically, 16× is used.

Note the 1:4 system clock/serial clock ratio does not apply when the DPLL is used to recover the clock in the 8×, 16×, or 32× modes. Synchronization occurs internally after the DPLL generates the Rx clock. Therefore, even the fastest DPLL clock generation (the 8× option) easily meets the required 1:4 ratio clocking limit.

27.4.6.1 Encoding Data with a DPLL

Each SCC contains a DPLL unit that can be programmed to encode and decode the SCC data as NRZ, NRZI Mark, NRZI Space, FM0, FM1, Manchester, and Differential Manchester. Figure 27-15 shows the different encoding methods.

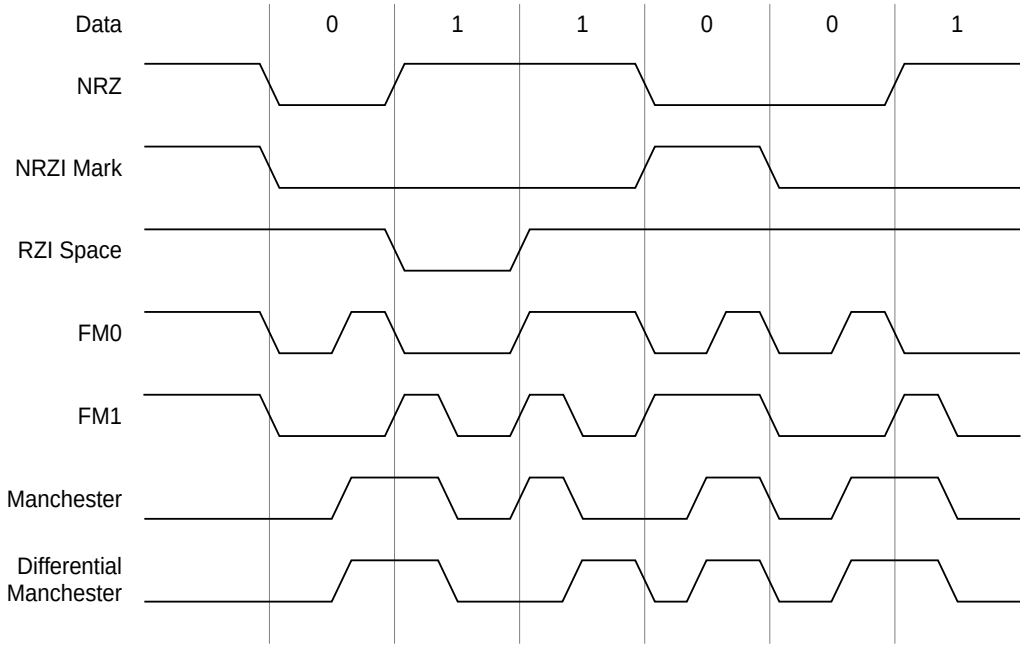


Figure 27-15. DPLL Encoding Examples

If the DPLL is not needed, NRZ or NRZI codings can be selected in GSMR_L[RENC, TENC]. Coding definitions are shown in Table 27-9.

Table 27-9. DPLL Codings

Coding	Description
NRZ	A one is represented by a high level for the duration of the bit and a zero is represented by a low level.
NRZI Mark	A one is represented by no transition at all. A zero is represented by a transition at the beginning of the bit (the level present in the preceding bit is reversed).
NRZI Space	A one is represented by a transition at the beginning of the bit (the level present in the preceding bit is reversed). A zero is represented by no transition at all.
FM0	A one is represented by a transition only at the beginning of the bit. A zero is represented by a transition at the beginning of the bit and another transition at the center of the bit.
FM1	A one is represented by a transition at the beginning of the bit and another transition at the center of the bit. A zero is represented by a transition only at the beginning of the bit.
Manchester	A one is represented by a high-to-low transition at the center of the bit. A zero is represented by a low to high transition at the center of the bit. In both cases there may be a transition at the beginning of the bit to set up the level required to make the correct center transition.
Differential Manchester	A one is represented by a transition at the center of the bit with the opposite direction from the transition at the center of the preceding bit. A zero is represented by a transition at the center of the bit with the same polarity from the transition at the center of the preceding bit.

27.4.7 Clock Glitch Detection

Clock glitches cause problems for many communications systems, and they may go undetected by the system. Systems that supply an external clock to a serial channel are often susceptible to glitches from noise, connecting or disconnecting the physical cable from the application board, or excessive ringing on a clock line. A clock glitch occurs when more than one edge occurs in a time period that violates the minimum high or low time specification of the input clock.

The SCCs on the MPC8560 have a special circuit designed to detect glitches and alert the system of a problem at the physical layer. The glitch-detect circuit is not a specification test; if a circuit does not meet the SCC's input clocking specifications, erroneous data may not be detected or false glitch indications can occur. Regardless of whether the DPLL is used, the received clock is passed through a noise filter that eliminates any noise spikes that affect a single sample. This sampling is enabled using GSMR_H[GDE].

If a spike is detected, a maskable Rx or Tx glitched clock interrupt is generated in SCCE*x*[GLR,GLT]. Although the receiver or transmitter can be reset or allowed to continue operation, statistics on clock glitches should be kept for evaluation to help in debugging, especially during prototype testing.

27.4.8 Reconfiguring the SCCs

The proper reconfiguration sequence must be followed for SCC parameters that cannot be changed dynamically. For instance, the internal baud rate generators allow on-the-fly changes, but the DPLL-related GSMR does not. The steps in the following sections show how to disable, reconfigure and re-enable an SCC to ensure that buffers currently in use are properly closed before reconfiguring the SCC and that subsequent data goes to or from new buffers according to the new configuration.

Modifying parameter RAM does not require the SCC to be fully disabled. See the parameter RAM description for when values can be changed. To disable all peripheral controllers, set CPCR[RST] to reset the entire CPM.

27.4.8.1 General Reconfiguration Sequence for an SCC Transmitter

An SCC transmitter can be reconfigured by following these general steps:

1. If the SCC is sending data, issue a STOP TRANSMIT command. Transmission should stop smoothly. If the SCC is not transmitting (no TxBDs are ready or the GRACEFUL STOP TRANSMIT command has been issued and completed) or the INIT TX PARAMETERS command is issued, the STOP TRANSMIT command is not required.
2. Clear GSMR_L[ENT] to disable the SCC transmitter and put it in reset state.
3. Modify SCC Tx parameters or parameter RAM. To switch protocols or restore the initial Tx parameters, issue an INIT TX PARAMETERS command.

4. If an INIT TX PARAMETERS command was not issued in step 3, issue a RESTART TRANSMIT command.
5. Set GSMR_L[ENT]. Transmission begins using the TxBD pointed to by TBPTR, assuming the R bit is set.

27.4.8.2 Reset Sequence for an SCC Transmitter

The following steps reinitialize an SCC transmit parameters to the reset state:

1. Clear GSMR_L[ENT].
2. Make any modifications then issue the INIT TX PARAMETERS command.
3. Set GSMR_L[ENT].

27.4.8.3 General Reconfiguration Sequence for an SCC Receiver

An SCC receiver can be reconfigured by following these steps:

1. Clear GSMR_L[ENR]. The SCC receiver is now disabled and put in a reset state.
2. Modify SCC Rx parameters or parameter RAM. To switch protocols or restore Rx parameters to their initial state, issue an INIT RX PARAMETERS command.
3. If the INIT RX PARAMETERS command was not issued in step 2, issue an ENTER HUNT MODE command.
4. Set GSMR_L[ENR]. Reception begins using the RxBD pointed to by RBPTR, assuming the E bit is set.

27.4.8.4 Reset Sequence for an SCC Receiver

To reinitialize the SCC receiver to the state it was in after reset, follow these steps:

1. Clear GSMR_L[ENR].
2. Make any modifications then issue the INIT RX PARAMETERS command.
3. Set GSMR_L[ENR].

27.4.8.5 Switching Protocols

To switch an SCC’s protocol without resetting the board or affecting other SCCs, follow these steps:

1. Clear GSMR_L[ENT, ENR].
2. Make protocol changes in the GSMR and additional parameters then issue the INIT TX and RX PARAMETERS command to initialize both Tx and Rx parameters.
3. Set GSMR_ L[ENT, ENR] to enable the SCC with the new protocol.

27.4.9 Saving Power

To save power when not in use, an SCC can be disabled by clearing GSMR_L[ENT, ENR].

Chapter 28
SCC UART Mode

The universal asynchronous receiver transmitter (UART) protocol is commonly used to send low-speed data between devices. The term asynchronous is used because it is not necessary to send clocking information along with the data being sent. UART links are typically 38,400 baud or less and are character-based. Asynchronous links are used to connect terminals with other devices. Even where synchronous communications are required, the UART is often used as a local port to run board debugger software. The character format of the UART protocol is shown in Figure 28-1.

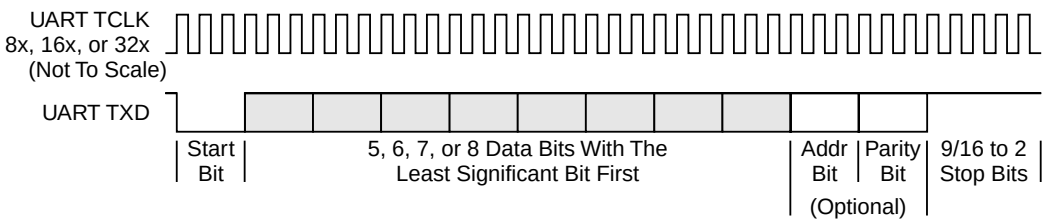


Figure 28-1. UART Character Format

Because the transmitter and receiver operate asynchronously, there is no need to connect the transmit and receive clocks. Instead, the receiver oversamples the incoming data stream (usually by a factor of 16) and uses some of these samples to determine the bit value. Traditionally, the middle 3 of the 16 samples are used. Two UARTs can communicate using this system if the transmitter and receiver use the same parameters, such as the parity scheme and character length.

When data is not sent, a continuous stream of ones is sent (idle condition). Because the start bit is always a zero, the receiver can detect when real data is once again on the line. UART specifies an all-zeros break character, which ends a character transfer sequence.

The most popular protocol that uses asynchronous characters is the RS-232 standard, which specifies baud rates, handshaking protocols, and mechanical/electrical details. Another popular format is RS-485, which defines a balanced line system allowing longer cables than RS-232 links. Even synchronous protocols like HDLC are sometimes defined to run over asynchronous links. The Profibus standard extends UART protocol to include LAN-oriented features such as token passing.

All standards provide handshaking signals, but some systems require only three physical lines—Tx data, Rx data, and ground. Many proprietary standards have been built around the UART's asynchronous character frame, some of which implement a multidrop configuration where multiple stations, each with a specific address, can be present on a network. In multidrop mode, frames of characters are broadcast with the first character acting as a destination address. To accommodate this, the UART frame is extended one bit to distinguish address characters from normal data characters.

In synchronous UART (isochronous operation), a separate clock signal is explicitly provided with the data. Start and stop bits are present in synchronous UART, but oversampling is not required because the clock is provided with each bit.

The general SCC mode register (GSMR) is used to configure an SCC channel to function in UART mode, which provides standard serial I/O using asynchronous character-based (start-stop) protocols with RS-232C-type lines. Using standard asynchronous bit rates and protocols, an SCC UART controller can communicate with any existing RS-232-type device and provides a serial communications port to other microprocessors and terminals (either locally or via modems). The independent transmit and receive sections, whose operations are asynchronous with the core, send data from memory (either internal or external) to TXD and receive data from RXD. The UART controller supports a multidrop mode for master/slave operations with wake-up capability on both the idle signal and address bit. It also supports synchronous operation where a clock (internal or external) must be provided with each bit received.

28.1 Features

The following list summarizes main features of an SCC UART controller:

- Flexible message-based data structure
- Implements synchronous and asynchronous UART
- Multidrop operation
- Receiver wake-up on idle line or address bit
- Receive entire messages into buffers as indicated by receiver idle timeout or by control character reception
- Eight control character comparison
- Two address comparison in multidrop configurations
- Maintenance of four 16-bit error counters
- Received break character length indication
- Programmable data length (5–8 bits)
- Programmable fractional stop bit lengths (from 9/16 to 2 bits) in transmission
- Capable of reception without a stop bit

- Even/odd/force/no parity generation and check
- Frame error, noise error, break, and idle detection
- Transmit preamble and break sequences
- Freeze transmission option with low-latency stop

28.2 Normal Asynchronous Mode

In normal asynchronous mode, the receive shift register receives incoming data on RXD*x*. Control bits in the UART mode register (PSMR) define the length and format of the UART character. Bits are received in the following order:

1. Start bit
2. 5–8 data bits (lsb first)
3. Address/data bit (optional)
4. Parity bit (optional)
5. Stop bits

The receiver uses a clock 8x 16x, or 32x faster than the baud rate and samples each bit of the incoming data three times around its center. The value of the bit is determined by the majority of those samples; if all do not agree, the noise indication counter (NOSEC) in parameter RAM is incremented. When a complete character has been clocked in, the contents of the receive shift register are transferred to the receive FIFO before proceeding to the receive buffer. The CPM flags UART events, including reception errors, in SCCE and the RxBD status and control fields.

The SCC can receive fractional stop bits. The next character's start bit can begin any time after the three middle samples are taken. The UART transmit shift register sends outgoing data on TXD*x*. Data is then clocked synchronously with the transmit clock, which may have either an internal or external source. Characters are sent lsb first. Only the data portion of the UART frame is stored in the buffers because start and stop bits are generated and stripped by the SCC. A parity bit can be generated in transmission and checked during reception; although it is not stored in the buffer, its value can be inferred from the buffer's reporting mechanism. Similarly, the optional address bit is not stored in the transmit or receive buffer, but is supplied in the BD itself. Parity generation and checking includes the optional address bit. GSMR_H[RFW] must be set for an 8-bit receive FIFO in the UART receiver.

28.3 Synchronous Mode

In synchronous mode, the controller uses a 1x data clock for timing. The receive shift register receives incoming data on RXD*x* synchronous with the clock. The bit length and format of the serial character are defined by the control bits in the PSMR in the same way

as in asynchronous mode. When a complete byte has been clocked in, the contents of the receive shift register are transferred to the receive FIFO before proceeding to the receive buffer. The CPM flags UART events, including reception errors, in SCCE and the RxBD status and control fields. GSMR_H[RFW] must be set for an 8-bit receive FIFO.

The synchronous UART transmit shift register sends outgoing data on TXD*x*. Data is then clocked synchronously with the transmit clock, which can have an internal or external source.

28.4 SCC UART Parameter RAM

For UART mode, the protocol-specific area of the SCC parameter RAM is mapped as in Table 28-1.

Table 28-1. UART-Specific SCC Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x30	—	DWord	Reserved
0x38	MAX_IDL	Hword	Maximum idle characters. When a character is received, the receiver begins counting idle characters. If MAX_IDL idle characters are received before the next data character, an idle timeout occurs and the buffer is closed, generating a maskable interrupt request to the core to receive the data from the buffer. Thus, MAX_IDL offers a way to demarcate frames. To disable the feature, clear MAX_IDL. The bit length of an idle character is calculated as follows: 1 + data length (5–9) + 1 (if parity is used) + number of stop bits (1–2). For 8 data bits, no parity, and 1 stop bit, the character length is 10 bits.
0x3A	IDLC	Hword	Temporary idle counter. Holds the current idle count for the idle timeout process. IDLC is a down-counter and does not need to be initialized or accessed.
0x3C	BRKCR	Hword	Break count register (transmit). Determines the number of break characters the transmitter sends. The transmitter sends a break character sequence when a STOP TRANSMIT command is issued. For 8 data bits, no parity, 1 stop bit, and 1 start bit, each break character consists of 10 zero bits.
0x3E	PAREC	Hword	User-initialized,16-bit (modulo–2^{16}) counters incremented by the CP. PAREC counts received parity errors. FRMEC counts received characters with framing errors. NOSEC counts received characters with noise errors. BRKEC counts break conditions on the signal. A break condition can last for hundreds of bit times, yet BRKEC is incremented only once during that period.
0x46	BRKLN	Hword	Last received break length. Holds the length of the last received break character sequence measured in character units. For example, if RXD*x* is low for 20 bit times and the defined character length is 10 bits, BRKLN = 0 × 002, indicating that the break sequence is at least 2 characters long. BRKLN is accurate to within one character length.
0x48	UADDR1	Hword	UART address character 1/2. In multidrop mode, the receiver provides automatic address recognition for two addresses. In this case, program the lower order bytes of UADDR1 and UADDR2 with the two preferred addresses.
0x4A	UADDR2	Hword	

Table 28-1. UART-Specific SCC Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x50	CHARACTER1	Hword	Control character 1–8. These characters define the Rx control characters on which interrupts can be generated.
0x52	CHARACTER2	Hword	
0x54	CHARACTER3	Hword	
0x56	CHARACTER4	Hword	
0x58	CHARACTER5	Hword	
0x5A	CHARACTER6	Hword	
0x5C	CHARACTER7	Hword	
0x5E	CHARACTER8	Hword	
0x60	RCCM	Hword	Receive control character mask. Used to mask comparison of CHARACTER1–8 so classes of control characters can be defined. A one enables the comparison, and a zero masks it.
0x62	RCCR	Hword	Receive control character register. Used to hold the last rejected control character (not written to the Rx buffer). Generates a maskable interrupt. If the core does not process the interrupt and read RCCR before a new control character arrives, the previous control character is overwritten.
0x64	RLBC	Hword	Receive last break character. Used in synchronous UART when PSMR[RZS] = 1; holds the last break character pattern. By counting zeros in RLBC, the core can measure break length to a one-bit resolution. Read RLBC by counting the zeros written from bit 0 to where the first one was written. RLBC = 0b001xxxxxxxxxxxx indicates two zeros; 0b1xxxxxxxxxxxxxx indicates no zeros. Note that RLBC can be used in combination with BRKLN above to calculate the number of bits in the break sequence: (BRKLN × character length) + (number of zeros in RLBC).

[1] From SCC base. See Section 27.4.1, “SCC Base Addresses.”

28.5 Data-Handling Methods: Character- or Message-Based

An SCC UART controller uses the same BD table and buffer structures as the other protocols and supports both multibuffer, message-based and single-buffer, character-based operation.

For character-based transfers, each character is sent with stop bits and parity and received into separate 1-byte buffers. A maskable interrupt is generated when each buffer is received.

In a message-based environment, transfers can be made on entire messages rather than on individual characters. To simplify programming and save processor overhead, a message is transferred as a linked list of buffers without core intervention. For example, before handling input data, a terminal driver may wait for an end-of-line character or an idle timeout rather than be interrupted when each character is received. Conversely, ASCII files can be sent as messages ending with an end-of-line character.

When receiving messages, up to eight control characters can be configured to mark the end of a message or generate a maskable interrupt without being stored in the buffer. This option is useful when flow control characters such as XON or XOFF are needed but are not part of the received message. See Section 28.9, “Receiving Control Characters.”

28.6 Error and Status Reporting

Overrun, parity, noise, and framing errors are reported via the BDs and/or error counters in the UART parameter RAM. Signal status is indicated in the status register; a maskable interrupt is generated when status changes.

28.7 SCC UART Commands

The transmit commands in Table 28-2 are issued to the CP command register (CPCR).

Table 28-2. Transmit Commands

Command	Description
STOP TRANSMIT	After a hardware or software reset and a channel is enabled in the GSMR, the transmitter starts polling the first BD in the TxBD table every 8 Tx clocks. STOP TRANSMIT disables character transmission. If the SCC receives STOP TRANSMIT as a message is being sent, the message is aborted. The transmitter finishes sending data transferred to its FIFO and stops. The TBPTR is not advanced. The UART transmitter sends a programmable break sequence and starts sending idles. The number of break characters in the sequence (which can be zero) should be written to BRKCR in the parameter RAM before issuing this command.
GRACEFUL STOP TRANSMIT	Used to stop transmitting smoothly. The transmitter stops after the current buffer has been completely sent or immediately if no buffer is being sent. SCCE[GRA] is set once transmission stops, then the UART Tx parameters, including the TxBD, can be modified. TBPTR points to the next TxBD in the table. Transmission begins once the R bit of the next BD is set and a RESTART TRANSMIT command is issued.
RESTART TRANSMIT	Enables transmission. The controller expects this command after it disables the channel in its PSMR, after a STOP TRANSMIT command, after a GRACEFUL STOP TRANSMIT command, or after a transmitter error. Transmission resumes from the current BD.
INIT TX PARAMETERS	Resets the transmit parameters in the parameter RAM. Issue only when the transmitter is disabled. Note that INIT TX AND RX PARAMETERS resets both Tx and Rx parameters.

Receive commands are described in Table 28-3.

Table 28-3. Receive Commands

Command	Description
ENTER HUNT MODE	Forces the receiver to close the RxBD in use and enter hunt mode. After a hardware or software reset, once an SCC is enabled in the GSMR, the receiver is automatically enabled and uses the first BD in the RxBD table. If a message is in progress, the receiver continues receiving in the next BD. In multidrop hunt mode, the receiver continually scans the input data stream for the address character. When it is not in multidrop mode, it waits for the idle sequence (one character of idle). Data present in the Rx FIFO is not lost when this command is executed.

Table 28-3. Receive Commands (continued)

Command	Description
CLOSE RXBD	Forces the SCC to close the RxBD in use and use the next BD for subsequent received data. If the SCC is not in the process of receiving data, no action is taken. Note that in an SCC UART controller, CLOSE RXBD functions like ENTER HUNT MODE but does not need to receive an idle character to continue receiving.
INIT RX PARAMETERS	Resets the receive parameters in the parameter RAM. Should be issued when the receiver is disabled. Note that INIT TX AND RX PARAMETERS resets both Tx and Rx parameters.

28.8 Multidrop Systems and Address Recognition

In multidrop systems, more than two stations can be on a network, each with a specific address. Figure 28-2. shows two examples of this configuration. Frames made up of many characters can be broadcast as long as the first character is the destination address. The UART frame is extended by one bit to distinguish an address character from standard data characters. Programmed in PSMR[UM], the controller supports the following two multidrop modes:

- Automatic multidrop mode—The controller checks the incoming address character and accepts subsequent data only if the address matches one of two user-defined values. The two 16-bit address registers, UADDR1 and UADDR2, support address recognition. Only the lower 8 bits are used so the upper 8 bits should be cleared; for addresses less than 8 bits, unused high-order bits should also be cleared. The incoming address is checked against UADDR1 and UADDR2. When a match occurs, RxBD[AM] indicates whether UADDR1 or UADDR2 matched.
- Manual multidrop mode—The controller receives all characters. An address character is always written to a new buffer and can be followed by data characters. User software performs the address comparison.

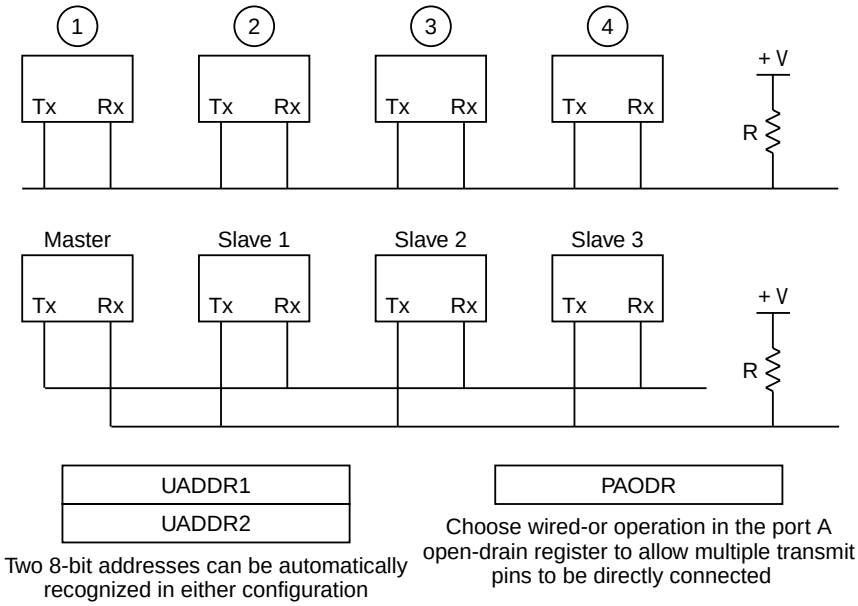


Figure 28-2. Two UART Multidrop Configurations

28.9 Receiving Control Characters

The UART receiver can recognize special control characters used in a message-based environment. Eight control characters can be defined in a control character table in the UART parameter RAM. Each incoming character is compared to the table entries using a mask (the received control character mask, RCCM) to strip don't cares. If a match occurs, the received control character can either be written to the receive buffer or rejected.

If the received control character is not rejected, it is written to the receive buffer. The receive buffer is then automatically closed to allow software to handle end-of-message characters. Control characters that are not part of the actual message, such as XOFF, can be rejected. Rejected characters bypass the receive buffer and are written directly to the received control character register (RCCR), which triggers maskable interrupt.

The 16-bit entries in the control character table support control character recognition. Each entry consists of the control character, a valid bit (end of table), and a reject bit. See Figure 28-3.

Offset [1]	0	1	2 – 7	8 – 15
0x50	E	R	—	CHARACTER1
0x52	E	R	—	CHARACTER2
⋮	⋮	⋮	⋮	⋮
0x5E	E	R	—	CHARACTER8
0x60	1	1	—	RCCM
0x62	—			RCCR

[1] From SCC*x* base address

Figure 28-3. Control Character Table

Table 28-4 describes the data structure used in control character recognition.

Table 28-4. Control Character Table, RCCM, and RCCR Descriptions

Offset	Bits	Name	Description
0x50–0x5E	0	E	End of table. In tables with eight control characters, E is always 0. 0 This entry is valid. 1 The entry is not valid and is not used.
	1	R	Reject character. 0 A matching character is not rejected but is written into the Rx buffer, which is then closed. If RxBD[I] is set, the buffer closing generates a maskable interrupt through SCCE[RX]. A new buffer is opened if more data is in the message. 1 A matching character is written to RCCR and not to the Rx buffer. A maskable interrupt is generated through SCCE[CCR]. The current Rx buffer is not closed.
	2–7	—	Reserved
	8–15	CHARACTERn	Control character values 1–8. Defines control characters to be compared to the incoming character. For characters smaller than 8 bits, the most significant bits should be zero.
0x60	0–1	0b11	Must be set. Used to mark the end of the control character table in case eight characters are used. Setting these bits ensures correct operation during control character recognition.
	2–7	—	Reserved
	8–15	RCCM	Received control character mask. Used to mask the comparison of CHARACTER*n*. Each RCCM bit corresponds to the respective bit of CHARACTER*n* and decodes as follows. 0 Ignore this bit when comparing the incoming character to CHARACTER*n*. 1 Use this bit when comparing the incoming character to CHARACTER*n*.
0x62	0–7	—	Reserved
	8–15	RCCR	Received control character register. If the newly arrived character matches and is rejected from the buffer (R = 1), the PIP controller writes the character into the RCCR and generates a maskable interrupt. If the core does not process the interrupt and read RCCR before a new control character arrives, the previous control character is overwritten.

28.10 Hunt Mode (Receiver)

A UART receiver in hunt mode remains deactivated until an idle or address character is recognized, depending on PSMR[UM]. A receiver is forced into hunt mode by issuing an ENTER HUNT MODE command.

The receiver aborts any message in progress when ENTER HUNT MODE is issued. When the message is finished, the receiver is reenabled by detecting the idle line (one idle character) or by the address bit of the next message, depending on PSMR[UM]. When a receiver in hunt mode receives a break sequence, it increments BRKEC and generates a BRK interrupt condition.

28.11 Inserting Control Characters into the Transmit Data Stream

The SCC UART transmitter can send out-of-sequence, flow-control characters like XON and XOFF. The controller polls the transmit out-of-sequence register (TOSEQ), shown in Figure 28-4, whenever the transmitter is enabled for UART operation, including during a UART freeze operation, UART buffer transmission, and when no buffer is ready for transmission. The TOSEQ character (in CHARSEND) is sent at a higher priority than the other characters in the transmit buffer, but does not preempt characters already in the transmit FIFO. This means that the XON or XOFF character may not be sent for eight or four (SCC) character times. To reduce this latency, set GSMR_H[TFL] to decrease the FIFO size to one character before enabling the transmitter.

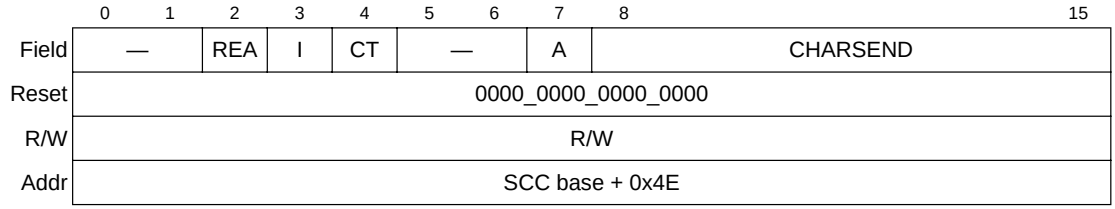


Figure 28-4. Transmit Out-of-Sequence Register (TOSEQ)

Table 28-5 describes TOSEQ fields.

Table 28-5. TOSEQ Field Descriptions

Bit	Name	Description
0–1	—	Reserved, should be cleared.
2	REA	Ready. Set when the character is ready for transmission. Remains 1 while the character is being sent. The CP clears this bit after transmission.
3	I	Interrupt. If this bit is set, transmission completion is flagged in the event register (SCCE[TX] is set), triggering a maskable interrupt to the core.

Table 28-5. TOSEQ Field Descriptions (continued)

Bit	Name	Description
4	CT	Clear-to-send lost. Operates only if the SCC monitors $\overline{\text{CTS}}$ (GSMR_L[DIAG]). The CP sets this bit if $\overline{\text{CTS}}$ negates when the TOSEQ character is sent. If $\overline{\text{CTS}}$ negates and the TOSEQ character is sent during a buffer transmission, the TxBD[CT] status bit is also set.
5–6	—	Reserved, should be cleared.
7	A	Address. Setting this bit indicates an address character for multidrop mode.
8–15	CHARSEND	Character send. Contains the character to be sent. Any 5- to 8-bit character value can be sent in accordance with the UART configuration. The character should be placed in the lsbs of CHARSEND. This value can be changed only while REA = 0.

28.12 Sending a Break (Transmitter)

A break is an all-zeros character with no stop bit that is sent by issuing a STOP TRANSMIT command. The SCC finishes transmitting outstanding data, sends a programmable number of break characters (determined by BRKCR), and reverts to idle or sends data if a RESTART TRANSMIT command is given before completion. When the break code is complete, the transmitter sends at least one high bit before sending more data, to guarantee recognition of a valid start bit. Because break characters do not preempt characters in the transmit FIFO, they may not be sent for eight (SCC) or four (SCC) character times. To reduce this latency, set GSMR_H[TFL] to decrease the FIFO size to one character before enabling the transmitter.

28.13 Sending a Preamble (Transmitter)

Sending a preamble sequence of consecutive ones ensures that a line is idle before sending a message. If the preamble bit TxBD[P] is set, the SCC sends a preamble sequence (idle character) before sending the buffer. For example, for 8 data bits, no parity, 1 stop bit, and 1 start bit, a preamble of 10 ones is sent before the first character in the buffer.

28.14 Fractional Stop Bits (Transmitter)

The asynchronous UART transmitter, shown in Figure 28-5., can be programmed to send fractional stop bits. The FSB field in the data synchronization register (DSR) determines the fractional length of the last stop bit to be sent. FSB can be modified at any time. If two stop bits are sent, only the second is affected. Idle characters are always sent as full-length characters.

	0	1 4	5	6	7	8	9	10	11	12	13	14	15
Field	—	FSB	—	—	—	—	—	—	—	—	—	—	—
Reset	0	1111	1	1	0	0	1	1	1	1	1	1	0
R/W	R/W												
Addr													

Figure 28-5. Asynchronous UART Transmitter

Table 28-6 describes DSR fields.

Table 28-6. DSR Fields Descriptions

Bit	Name	Description
0	—	0b0
1–4	FSB	Fractional stop bits. For 16× oversampling: 1111 Last transmitted stop bit 16/16. Default value after reset. 1110 Last transmitted stop bit 15/16 …. 1000 Last transmitted stop bit 9/16. 0xxx Invalid. Do not use. For 32× oversampling: 1111 Last transmitted stop bit 32/32. Default value after reset. 1110 Last transmitted stop bit 31/32. …. 0000 Last transmitted stop bit 17/32. For 8× oversampling: 1111 Last transmitted stop bit 8/8. Default value after reset. 1110 Last transmitted stop bit 7/8. 1101 Last transmitted stop bit 6/8. 1100 Last transmitted stop bit 5/8. 10xx Invalid. Do not use. 0xxx Invalid. Do not use. The UART receiver can always receive fractional stop bits. The next character's start bit can begin any time after the three middle samples have been taken.
5–6	—	0b11
7–8	—	0b00
9–14	—	0b111111
15	—	0b0

28.15 Handling Errors in the SCC UART Controller

The UART controller reports character reception and transmission error conditions via the BDs, the error counters, and the SCCE. Modem interface lines can be monitored by the port C pins. Transmission errors are described in Table 28-7.

Table 28-7. Transmission Errors

Error	Description
$\overline{\text{CTS}}$ Lost during Character Transmission	When $\overline{\text{CTS}}$ negates during transmission, the channel stops after finishing the current character. The CP sets TxBD[CT] and generates the TX interrupt if it is not masked. The channel resumes transmission after the RESTART TRANSMIT command is issued and $\overline{\text{CTS}}$ is asserted. Note that if $\overline{\text{CTS}}$ is used, the UART also offers an asynchronous flow control option that does not generate an error. See the description of PSMR[FLC] in Table 28-9.

Reception errors are described in Table 28-8.

Table 28-8. Reception Errors

Error	Description
Overrun	Occurs when the channel overwrites the previous character in the Rx FIFO with a new character, losing the previous character. The channel then writes the new character to the buffer, closes it, sets RxBD[OV], and generates an RX interrupt if not masked. In automatic multidrop mode, the receiver enters hunt mode immediately.
$\overline{\text{CD}}$ Lost during Character Reception	If this error occurs and the channel is using this pin to automatically control reception, the channel terminates character reception, closes the buffer, sets RxBD[CD], and generates the RX interrupt if not masked. This error has the highest priority. The last character in the buffer is lost and other errors are not checked. In automatic multidrop mode, the receiver enters the hunt mode immediately.
Parity	When a parity error occurs, the channel writes the received character to the buffer, closes the buffer, sets RxBD[PR], and generates the RX interrupt if not masked. The channel also increments the parity error counter PAREC. In automatic multidrop mode, the receiver enters hunt mode immediately.
Noise	A noise error occurs when the three samples of a bit are not identical. When this error occurs, the channel writes the received character to the buffer, proceeds normally, but increments the noise error counter NOSEC. Note that this error does not occur in synchronous mode.
Idle Sequence Receive	If the UART is receiving data and gets an idle character (all ones), the channel begins counting consecutive idle characters received. If MAX_IDL is reached, the buffer is closed and an RX interrupt is generated if not masked. If no buffer is open, this event does not generate an interrupt or any status information. The internal idle counter (IDLC) is reset every time a character is received. To disable the idle sequence function, clear MAX_IDL.
Framing	The UART reports a framing errors when it receives a character with no stop bit, regardless of the mode. The channel writes the received character to the buffer, closes it, sets RxBD[FR], generates the RX interrupt if not masked, increments FRMEC, but does not check parity for this character. In automatic multidrop mode, the receiver immediately enters hunt mode. If the UART allows data with no stop bits (PSMR[RZS] = 1) when in synchronous mode (PSMR[SYN] = 1), framing errors are reported but reception continues assuming the unexpected zero is the start bit of the next character; in this case, the user may ignore a reported framing error until multiple framing errors occur within a short period.
Break Sequence	When the first break sequence is received, the UART increments the break error counter BRKEC. It updates BRKLN when the sequence completes. After the first 1 is received, the UART sets SCCE[BRKE], which generates an interrupt if not masked. If the UART is receiving characters when it receives a break, it closes the Rx buffer, sets RxBD[BR], and sets SCCE[RX], which can generate an interrupt if not masked. If PSMR[RZS] = 1 when the UART is in synchronous mode, a break sequence is detected after two successive break characters are received.

28.16 UART Mode Register (PSMR)

For UART mode, the SCC protocol-specific mode register (PSMR) is called the UART mode register. Many bits can be modified while the receiver and transmitter are enabled. Figure 28-6 shows the PSMR in UART mode.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	FLC	SL	CL		UM		FRZ	RZS	SYN	DRT	—	PEN	RPM		TPM	
Reset	0															
R/W	R/W															
Addr	0x9_1A08 (PSMR1); 0x9_1A28 (PSMR2); 0x9_1A48 (PSMR3); 0x9_1A68 (PSMR4)															

Figure 28-6. Protocol-Specific Mode Register for UART (PSMR)

Table 28-9 describes PSMR UART fields.

Table 28-9. PSMR UART Field Descriptions

Bit	Name	Description
0	FLC	Flow control. 0 Normal operation. The GSMR and port C registers determine the mode of $\overline{\text{CTS}}$. 1 Asynchronous flow control. When $\overline{\text{CTS}}$ is negated, the transmitter stops at the end of the current character. If $\overline{\text{CTS}}$ is negated past the middle of the current character, the next full character is sent before transmission stops. When $\overline{\text{CTS}}$ is asserted again, transmission continues where it left off and no $\overline{\text{CTS}}$ lost error is reported. Only idle characters are sent while $\overline{\text{CTS}}$ is negated.
1	SL	Stop length. Selects the number of stop bits the SCC sends. SL can be modified on-the-fly. The receiver is always enabled for one stop bit unless the SCC UART is in synchronous mode and PSMR[RZS] is set. Fractional stop bits are configured in the DSR. 0 1stop bit 1 2 stop bits
2–3	CL	Character length. Determines the number of data bits in the character, not including optional parity or multidrop address bits. If a character is less than 8 bits, most-significant bits are received as zeros and are ignored when the character is sent. CL can be modified on-the-fly. 00 5 data bits 01 6 data bits 10 7 data bits 11 8 data bits
4–5	UM	UART mode. Selects the asynchronous channel protocol. UM can be modified on-the-fly. 00 Normal UART operation. Multidrop mode is disabled and idle-line wake-up mode is selected. The UART receiver leaves hunt mode by receiving an idle character (all ones). 01 Manual multidrop mode. An additional address/data bit is sent with each character. Multidrop asynchronous modes are compatible with the MC68681 DUART, MC68HC11 SCI, DSP56000 SCI, and Intel 8051 serial interface. The receiver leaves hunt mode when the address/data bit is a one, indicating the received character is an address that all inactive processors must process. The controller receives the address character and writes it to a new buffer. The core then compares the written address with its own address and decides whether to ignore or process subsequent characters. 10 Reserved. 11 Automatic multidrop mode. The CPM compares the address of an incoming address character with UADDR*x* parameter RAM values; subsequent data is accepted only if a match occurs.

Table 28-9. PSMR UART Field Descriptions (continued)

Bit	Name	Description
6	FRZ	Freeze transmission. Allows the UART transmitter to pause and later continue from that point. 0 Normal operation. If the buffer was previously frozen, it resumes transmission from the next character in the same buffer that was frozen. 1 The SCC completes transmission of any data already transferred to the Tx FIFO (the number of characters depends on GSMR_H[TFL]) and then freezes. After FRZ is cleared, transmission resumes from the next character.
7	RZS	Receive zero stop bits. 0 The receiver operates normally, but at least one stop bit is needed between characters. A framing error is issued if a stop bit is missing. Break status is set if an all-zero character is received with a zero stop bit. 1 Configures the receiver to receive data without stop bits. Useful in V.14 applications where SCC UART controller data is supplied synchronously and all stop bits of a particular character can be omitted for cross-network rate adaptation. RZS should be set only if SYN is set. The receiver continues if a stop bit is missing. If the stop bit is a zero, the next bit is considered the first data bit of the next character. A framing error is issued if a stop bit is missing, but a break status is reported only after two consecutive break characters have no stop bits.
8	SYN	Synchronous mode. 0 Normal asynchronous operation. GSMR_L[TENC,RENC] must select NRZ and GSMR_L[TDCR, RDCR] select either 8×, 16×, or 32×. 16× is recommended for most applications. 1 Synchronous SCC UART controller using 1× clock (isochronous UART operation). GSMR_L[TENC, RENC] must select NRZ and GSMR_L[RDCR, TDCR] select 1× mode. A bit is transferred with each clock and is synchronous to the clock, which can be internal or external.
9	DRT	Disable receiver while transmitting. 0 Normal operation. 1 While the SCC is sending data, the internal $\overline{\text{RTS}}$ disables and gates the receiver. Useful for a multidrop configuration in which the user does not want to receive its own transmission. For multidrop UART mode, set the BDs' preamble bit, TxBD[P].
10	—	Reserved, should be cleared.
11	PEN	Parity enable. 0 No parity. 1 Parity is enabled and determined by the parity mode bits.
12–13, 14–15	RPM, TPM	Receiver/transmitter parity mode. Selects the type of parity check the receiver/transmitter performs; can be modified on-the-fly. Receive parity errors can be ignored but not disabled. 00 Odd parity. If a transmitter counts an even number of ones in the data word, it sets the parity bit so an odd number is sent. If a receiver receives an even number, a parity error is reported. 01 Low parity (space parity). A transmitter sends a zero in the parity bit position. If a receiver does not read a 0 in the parity bit, a parity error is reported. 10 Even parity. Like odd parity, the transmitter adjusts the parity bit, as necessary, to ensure that the receiver receives an even number of one bits; otherwise, a parity error is reported. 11 High parity (mark parity). The transmitter sends a one in the parity bit position. If the receiver does not read a 1 in the parity bit, a parity error is reported.

28.17 SCC UART Receive Buffer Descriptor (RxBD)

The CPM uses RxBDs to report on each buffer received. The CPM closes the current buffer, generates a maskable interrupt, and starts receiving data into the next buffer after one of the following occurs:

- A user-defined control character is received.
- An error occurs during message processing.
- A full receive buffer is detected.
- A MAX_IDL number of consecutive idle characters is received.
- An ENTER HUNT MODE or CLOSE RXBD command is issued.
- An address character is received in multidrop mode. The address character is written to the next buffer for a software comparison.

Figure 28-7 shows an example of how RxBDs are used in receiving.

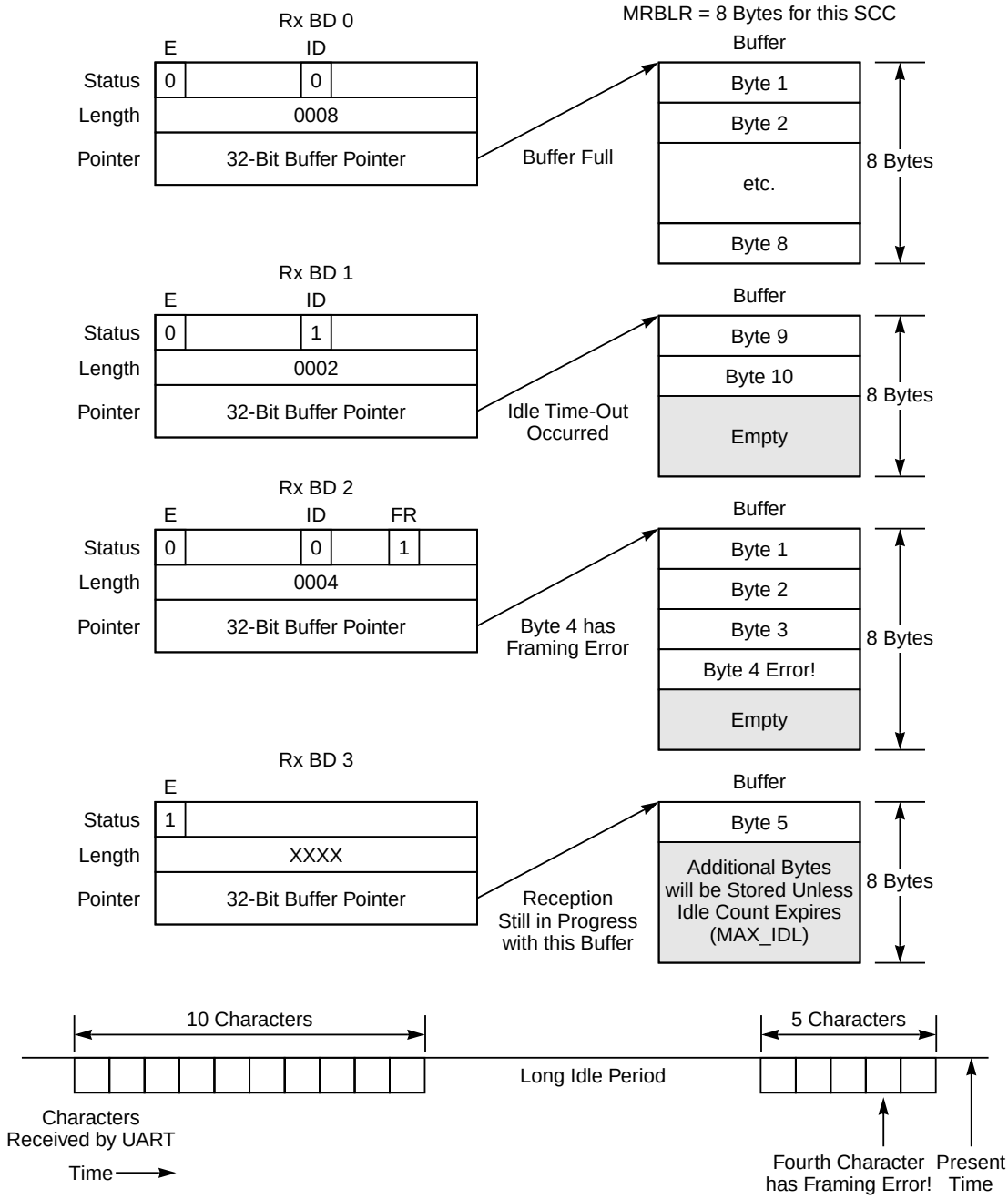


Figure 28-7. SCC UART Receiving using RxBDs

Figure 28-8 shows the SCC UART RxBD.

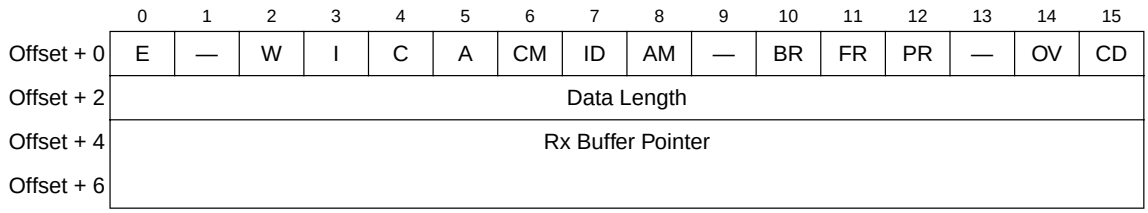


Figure 28-8. SCC UART Receive Buffer Descriptor (RxBD)

Table 28-10 describes RxBD status and control fields.

Table 28-10. SCC UART RxBD Status and Control Field Descriptions

Bits	Name	Description
0	E	Empty. 0 The buffer is full or reception was aborted due to an error. The core can read or write to any fields of this BD. The CPM does not reuse this BD while E = 0. 1 The buffer is not full. The CPM controls this BD and buffer. The core should not modify this BD.
1	—	Reserved, should be cleared.
2	W	Wrap (last buffer descriptor in the BD table). 0 Not the last descriptor in the table. 1 Last descriptor in the table. After this buffer is used, the CPM receives incoming data using the BD pointed to by RBASE. The number of BDs in this table is programable and determined only by the W bit and overall space constraints of the dual-port RAM.
3	I	Interrupt. 0 No interrupt is generated after this buffer is filled. 1 The CP sets SCCE[RX] when this buffer is completely filled by the CPM, indicating the need for the core to process the buffer. Setting SCCE[RX] causes an interrupt if not masked.
4	C	Control character. 0 This buffer does not contain a control character. 1 The last byte in this buffer matches a user-defined control character.
5	A	Address. 0 The buffer contains only data. 1 For manual multidrop mode, A indicates the first byte of this buffer is an address byte. Software should perform address comparison. In automatic multidrop mode, A indicates the buffer contains a message received immediately after an address matched UADDR1 or UADDR2. The address itself is not written to the buffer but is indicated by the AM bit.
6	CM	Continuous mode. 0 Normal operation. The CPM clears E after this BD is closed. 1 The CPM does not clear E after this BD is closed, allowing the buffer to be overwritten when the CPM accesses this BD again. E is cleared if an error occurs during reception, regardless of CM.
7	ID	Buffer closed on reception of idles. The buffer is closed because a programmable number of consecutive idle sequences (MAX_IDL) was received.
8	AM	Address match. Significant only if the address bit is set and automatic multidrop mode is selected in PSMR[UM]. After an address match, AM identifies which user-defined address character was matched. 0 The address matched the value in UADDR2. 1 The address matched the value in UADDR1.
9	—	Reserved, should be cleared.

Table 28-10. SCC UART RxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
10	BR	Break received. Set when a break sequence is received as data is being received into this buffer.
11	FR	Framing error. Set when a character with a framing error (a character without a stop bit) is received and located in the last byte of this buffer. A new Rx buffer is used to receive subsequent data.
12	PR	Parity error. Set when a character with a parity error is received and located in the last byte of this buffer. A new Rx buffer is used to receive subsequent data.
13	—	Reserved, should be cleared.
14	OV	Overrun. Set when a receiver overrun occurs during reception.
15	CD	Carrier detect lost. Set when the carrier detect signal is negated during reception.

Section 27.3, “SCC Buffer Descriptors (BDs),” describes the data length and buffer pointer fields.

28.18 SCC UART Transmit Buffer Descriptor (TxBD)

The CPM uses BDs to confirm transmission and indicate error conditions so the core knows that buffers have been serviced. Figure 28-9 shows the SCC UART TxBD.

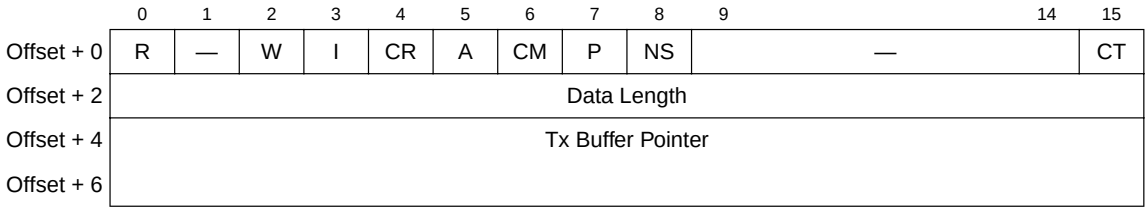


Figure 28-9. SCC UART Transmit Buffer Descriptor (TxBD)

Table 28-11 describes TxBD status and control fields.

Table 28-11. SCC UART TxBD Status and Control Field Descriptions

Bit	Name	Description
0	R	Ready. 0 The buffer is not ready. This BD and buffer can be modified. The CPM automatically clears R after the buffer is sent or an error occurs. 1 The user-prepared buffer is waiting to begin transmission or is being transmitted. Do not modify the BD once R is set.
1	—	Reserved, should be cleared.
2	W	Wrap (last buffer descriptor in TxBD table). 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CPM sends data using the BD pointed to by TBASE. The number of TxBDs in this table is determined only by the W bit and space constraints of the dual-port RAM.

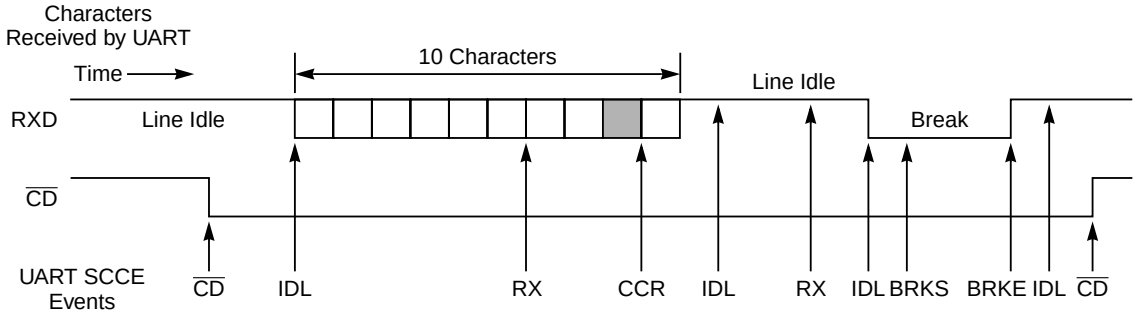
Table 28-11. SCC UART TxBD Status and Control Field Descriptions (continued)

Bit	Name	Description
3	I	Interrupt. 0 No interrupt is generated after this buffer is processed. 1 SCCE[TX] is set after this buffer is processed by the CPM, which can cause an interrupt.
4	CR	Clear-to-send report. 0 The next buffer is sent with no delay (assuming it is ready), but if a $\overline{\text{CTS}}$ lost condition occurs, TxBD[CT] may not be set in the correct TxBD or may not be set at all. Asynchronous flow control, however, continues to function normally. 1 Normal $\overline{\text{CTS}}$ lost error reporting and three bits of idle are sent between consecutive buffers.
5	A	Address. Valid only in multidrop mode—automatic or manual. 0 This buffer contains only data. 1 This buffer contains address characters. All data in this buffer is sent as address characters.
6	CM	Continuous mode. 0 Normal operation. The CPM clears R after this BD is closed. 1 The CPM does not clear R after this BD is closed, allowing the buffer to be resent next time the CPM accesses this BD. However, R is cleared by transmission errors, regardless of CM.
7	P	Preamble. 0 No preamble sequence is sent. 1 Before sending data, the controller sends an idle character consisting of all ones. If the data length of this BD is zero, only a preamble is sent.
8	NS	No stop bit or shaved stop bit sent. 0 Normal operation. Stop bits are sent with all characters in this buffer. 1 If PSMR[SYN] = 1, data in this buffer is sent without stop bits. If SYN = 0, the stop bit is shaved, depending on the DSR setting; see Section 28.14, “Fractional Stop Bits (Transmitter).”
9–14	—	Reserved, should be cleared.
15	CT	$\overline{\text{CTS}}$ lost. The CPM writes this status bit after sending the associated buffer. 0 $\overline{\text{CTS}}$ remained asserted during transmission. 1 $\overline{\text{CTS}}$ negated during transmission.

The data length and buffer pointer fields are described in Section 27.3, “SCC Buffer Descriptors (BDs).”

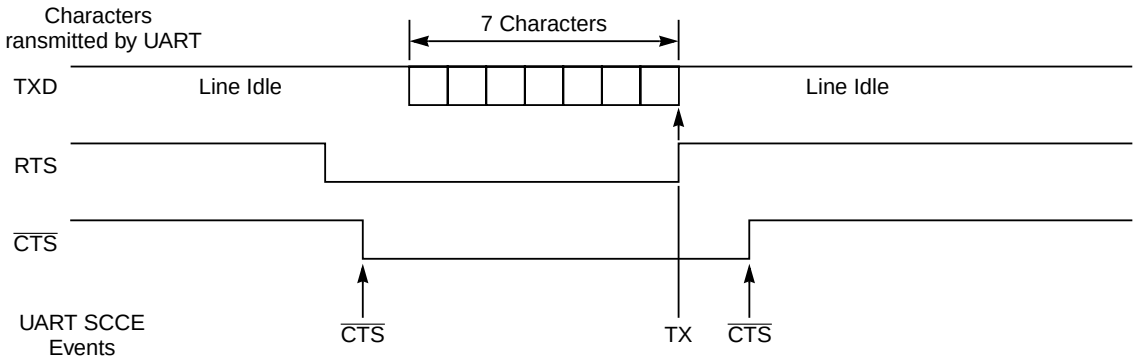
28.19 SCC UART Event Register (SCCE) and Mask Register (SCCM)

The SCC event register (SCCE) is used to report events recognized by the UART channel and to generate interrupts. When an event is recognized, the controller sets the corresponding SCCE bit. Interrupts can be masked in the UART mask register (SCCM), which has the same format as SCCE. Setting a mask bit enables the corresponding SCCE interrupt; clearing a bit masks it. Figure 28-10 shows example interrupts that can be generated by the SCC UART controller.



Notes:
1. The first RX event assumes Rx buffers are 6 bytes each.
2. The second IDL event occurs after an all-ones character is received.
3. The second RX event position is programmable based on the MAX_IDL value.
4. The BRKS event occurs after the first break character is received.
5. The CD event must be programmed in the port C parallel I/O, not in the SCC itself.

Legend:
A receive control character defined not to be stored in the Rx buffer.



Notes:
1. TX event assumes all seven characters were put into a single buffer and TxBD[CR]=1.
2. The CTS event must be programmed in the port C parallel I/O, not in the SCC itself.

Figure 28-10. SCC UART Interrupt Event Example

SCCE bits are cleared by writing ones; writing zeros has no effect. Unmasked bits must be cleared before the CPM clears an internal interrupt request. Figure 28-11 shows SCCE/SCCM for UART operation.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	—			GLR	GLT	—	AB	IDL	GRA	BRKE	BRKS	—	CCR	BSY	TX	RX
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_1A10 (SCCE1); 0x9_1A30 (SCCE2); 0x9_1A50 (SCCE3); 0x9_1A70 (SCCE4) 0x9_1A14 (SCCM1); 0x9_1A34 (SCCM2); 0x9_1A54 (SCCM3); 0x9_1A74 (SCCM4)															

Figure 28-11. SCC UART Event Register (SCCE) and Mask Register (SCCM)

Table 28-12 describes SCCE fields for UART mode.

Table 28-12. SCCE/SCCM Field Descriptions for UART Mode

Bit	Name	Description
0–2	—	Reserved, should be cleared.
3	GLR	Glitch on receive. Set when the SCC encounters an Rx clock glitch.
4	GLT	Glitch on transmit. Set when the SCC encounters a Tx clock glitch.
5	—	Reserved, should be cleared.
6	AB	Autobaud. Set when an autobaud lock is detected. The core should rewrite the baud rate generator with the precise divider value. See Chapter 24, “Baud-Rate Generators (BRGs).”
7	IDL	Idle sequence status changed. Set when the channel detects a change in the serial line. The line’s real-time status can be read in SCCS[ID]. Idle is entered when a character of all ones is received; it is exited when a zero is received.
8	GRA	Graceful stop complete. Set as soon as the transmitter finishes any buffer in progress after a GRACEFUL STOP TRANSMIT command is issued. It is set immediately if no buffer is in progress.
9	BRKE	Break end. Set when an idle bit is received after a break sequence.
10	BRKS	Break start. Set when the first character of a break sequence is received. Multiple BRKS events are not received if a long break sequence is received.
11	—	Reserved, should be cleared.
12	CCR	Control character received and rejected. Set when a control character is recognized and stored in the receive control character register RCCR.
13	BSY	Busy. Set when a character is received and discarded due to a lack of buffers. In multidrop mode, the receiver automatically enters hunt mode; otherwise, reception continues when a buffer is available. The latest point that an RxBD can be changed to empty and guarantee avoiding the busy condition is the middle of the stop bit of the first character to be stored in that buffer.
14	TX	Tx event. Set when a buffer is sent. If TxBD[CR] = 1, TX is set no sooner than when the last stop bit of the last character in the buffer begins transmission. If TxBD[CR] = 0, TX is set after the last character is written to the Tx FIFO. TX also represents a $\overline{\text{CTS}}$ lost error; check TxBD[CT].
15	RX	Rx event. Set when a buffer is received, which is no sooner than the middle of the first stop bit of the character that caused the buffer to close. Also represents a general receiver error (overrun, $\overline{\text{CD}}$ lost, parity, idle sequence, and framing errors); the RxBD status and control fields indicate the specific error.

28.20 SCC UART Status Register (SCCS)

The SCC UART status register (SCCS), shown in Figure 28-12, monitors the real-time status of RXD.

	0 – 6	7
Field	—	ID
Reset	0000_0000_0000_0000	
R/W	R	
Addr	0x9_1A17 (SCCS1); 0x9_1A37 (SCCS2); 0x9_1A57 (SCCS3); 0x9_1A77 (SCCS4)	

Figure 28-12. SCC Status Register for UART Mode (SCCS)

Table 28-13 describes UART SCCS fields.

Table 28-13. UART SCCS Field Descriptions

Bits	Name	Description
0–6	—	Reserved, should be cleared.
7	ID	Idle status. Set when RXD has been a logic one for at least a full character time. 0 The line is not idle. 1 The line is idle.

28.21 SCC UART Programming Example

The following initialization sequence is for the 9600 baud, 8 data bits, no parity, and stop bit of an SCC in UART mode assuming a 66-MHz system frequency. BRG1 and SCC2 are used. The controller is configured with $\overline{\text{RTS2}}$, $\overline{\text{CTS2}}$, and $\overline{\text{CD2}}$ active; $\overline{\text{CTS2}}$ acts as an automatic flow-control signal.

1. Configure port D pins to enable TXD2 and RXD2. Set PPARD[27,28] and PDIRD[27] and clear PDIRD[28] and PSORD[27,28].
2. Configure ports C and D pins to enable $\overline{\text{RTS2}}$, $\overline{\text{CTS2}}$ and $\overline{\text{CD2}}$. Set PPARD[26], PPARC[12,13] and PDIRD[26] and clear PDIRC[12,13], PSORC[12,13] and PSORD[26].
3. Configure BRG1. Write BRGC1 with 0x0001_035A. The DIV16 bit is not used and the divider is 429 (decimal). The resulting BRG1 clock is 16× the preferred bit rate.
4. Connect BRG1 to SCC2 using the CPM mux. Clear CMXSCR[RS2CS,TS2CS].
5. Connect the SCC2 to the NMSI. Clear CMXSCR[SC2].
6. Write RBASE and TBASE in the SCC2 parameter RAM to point to the RxBD and TxBD tables in dual-port RAM. Assuming one RxBD at the start of dual-port RAM followed by one TxBD, write RBASE with 0x0000 and TBASE with 0x0008.
7. Write 0x04A1_0000 to CPCR to execute the INIT RX AND TX PARAMS command for SCC2. This command updates RBPTR and TBPTR of the serial channel with the new values of RBASE and TBASE.
8. Write RFCR with 0x10 and TFCR with 0x10 for normal operation.

9. Write MRBLR with the maximum number of bytes per Rx buffer. For this case, assume 16 bytes, so MRBLR = 0x0010.
10. Write MAX_IDL with 0x0000 in the parameter RAM to disable the maximum idle functionality for this example.
11. Set BRKCR to 0x0001 so STOP TRANSMIT commands send only one break character.
12. Clear PAREC, FRMEC, NOSEC, and BRKEC in parameter RAM.
13. Clear UADDR1 and UADDR2. They are not used.
14. Clear TOSEQ. It is not used.
15. Write CHARACTER1–8 with 0x8000. They are not used.
16. Write RCCM with 0xC0FF. It is not used.
17. Initialize the RxBD. Assume the Rx buffer is at 0x0000_1000 in main memory. Write 0xB000 to the RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x0000_1000 to RxBD[Buffer Pointer].
18. Initialize the TxBD. Assume the buffer is at 0x0000_2000 in main memory and contains sixteen 8-bit characters. Write 0xB000 to the TxBD[Status and Control], 0x0010 to TxBD[Data Length], and 0x00002000 to TxBD[Buffer Pointer].
19. Write 0xFFFF to SCCE2 to clear any previous events.
20. Write 0x0003 to SCCM2 to allow the TX and RX interrupts.
21. Write 0x0040_0000 to the SIMR_L so SCC2 can generate a system interrupt. Initialize SIPNR_L by writing 0xFFFF_FFFF to it.
22. Write 0x0000_0020 to GSMR_H2 to configure a small Rx FIFO width.
23. Write 0x0002_8004 to GSMR_L2 to configure 16× sampling for transmit and receive, $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ to automatically control transmission and reception (DIAG bits), and the SCC for UART mode. Notice that the transmitter (ENT) and receiver (ENR) have not been enabled yet.
24. Set PSMR2 to 0xB000 to configure automatic flow control using $\overline{\text{CTS}}$, 8-bit characters, no parity, 1 stop bit, and asynchronous SCC UART operation.
25. Write 0x0002_8034 to GSMR_L2 to enable the transmitter and receiver. This ensures that ENT and ENR are enabled last.

Note that after 16 bytes are sent, the transmit buffer is closed. Additionally, the receive buffer is closed after 16 bytes are received. Data received after 16 bytes causes a busy (out-of-buffers) condition because only one RxBD is prepared.

28.22 S-Records Loader Application

This section describes a downloading application that uses an SCC UART controller. The application performs S-record downloads and uploads between a host computer and an

intelligent peripheral through a serial asynchronous line. S-records are strings of ASCII characters that begin with ‘S’ and end in an end-of-line character. This characteristic is used to impose a message structure on the communication between the devices. For flow control, each device can transmit XON and XOFF characters, which are not part of the program being uploaded or downloaded.

For simplicity, assume that the line is not multidrop (no addresses are sent) and that each S-record fits into a single buffer. Follow the basic UART initialization sequence above in Section 28.21, “SCC UART Programming Example,” except allow for more and larger buffers and create the control character table as described in Table 28-14.

Table 28-14. UART Control Characters for S-Records Example

Character	Description
Line Feed	Both the E and R bits should be cleared. When an end-of-line character is received, the current buffer is closed and made available to the core for processing. This buffer contains an entire S record that the processor can now check and copy to memory or disk as required.
XOFF	E should be cleared; R should be set. Whenever the core receives a control-character-received (CCR) interrupt and the RCCR contains XOFF, the software should immediately stop transmitting by setting PSMR[FRZ]. This keeps the other station from losing data when it runs out of Rx buffers.
XON	XON should be received after XOFF. E should be cleared and R should be set. PSMR[FRZ] on the transmitter should now be cleared. The CPM automatically resumes transmission of the serial line at the point at which it was previously stopped. Like XOFF, the XON character is not stored in the receive buffer.

To receive S-records, the core must wait for an RX interrupt, indicating that a complete S-record buffer was received. Transmission requires assembling S-records into buffers and linking them to the TxBD table; transmission can be paused when an XOFF character is received. This scheme minimizes the number of interrupts the core receives (one per S-record) and relieves it from continually scanning for control characters.

Chapter 29
SCC HDLC Mode

High-level data link control (HDLC) is one of the most common protocols in the data link layer, layer 2 of the OSI model. Many other common layer 2 protocols, such as SDLC, SS#7, AppleTalk, LAPB, and LAPD, are based on HDLC and its framing structure in particular. Figure 29-1 shows the HDLC framing structure.

HDLC uses a zero insertion/deletion process (bit-stuffing) to ensure that a data bit pattern matching the delimiter flag does not occur in a field between flags. The HDLC frame is synchronous and relies on the physical layer for clocking and synchronization of the transmitter/receiver.

An address field is needed to carry the frame's destination address because the layer 2 frame can be sent over point-to-point links, broadcast networks, packet-switched or circuit-switched systems. An address field is commonly 0, 8, or 16 bits, depending on the data link layer protocol. SDLC and LAPB use an 8-bit address. SS#7 has no address field because it is always used in point-to-point signaling links. LAPD divides its 16-bit address into different fields to specify various access points within one device. LAPD also defines a broadcast address. Some HDLC-type protocols permit addressing beyond 16 bits.

The 8- or 16-bit control field provides a flow control number and defines the frame type (control or data). The exact use and structure of this field depends on the protocol using the frame. The length of the data in the data field depends on the frame protocol. Layer 3 frames are carried in this data field. Error control is implemented by appending a cyclic redundancy check (CRC) to the frame, which in most protocols is 16 bits long but can be as long as 32 bits. In HDLC, the lsb of each octet is sent first; the msb of the CRC is sent first.

HDLC mode is selected for an SCC by writing GSMR_L[MODE] = 0b0000. In a nonmultiplexed modem interface, SCC outputs connect directly to external pins. Modem signals can be supported through port C. The Rx and Tx clocks can be supplied from either the bank of baud rate generators, by the DPLL, or externally. An SCC can also be connected through the TDM channels of the serial interface (SI). In HDLC mode, an SCC becomes an HDLC controller, and consists of separate transmit and receive sections whose operations are asynchronous with the core and can either be synchronous or asynchronous with respect to other SCCs.

29.1 SCC HDLC Features

The main features of an SCC in HDLC mode are follows:

- Flexible buffers with multiple buffers per frame
- Separate interrupts for frames and buffers (Rx and Tx)
- Received-frames threshold to reduce interrupt overhead
- Can be used with the SCC DPLL
- Four address comparison registers with mask
- Maintenance of five 16-bit error counters
- Flag/abort/idle generation and detection
- Zero insertion/deletion
- 16- or 32-bit CRC-CCITT generation and checking
- Detection of nonoctet aligned frames
- Detection of frames that are too long
- Programmable flags (0–15) between successive frames
- Automatic retransmission in case of collision

29.2 SCC HDLC Channel Frame Transmission

The HDLC transmitter is designed to work with little or no core intervention. Once enabled by the core, a transmitter starts sending flags or idles as programmed in the HDLC mode register (PSMR). The HDLC polls the first BD in the TxBD table. When there is a frame to transmit, the SCC fetches the data (address, control, and information) from the first buffer and starts sending the frame after inserting the minimum number of flags specified between frames. When the end of the current buffer is reached and TxBD[L] (last buffer in frame) is set, the SCC appends the CRC and closing flag. In HDLC mode, the lsb of each octet and the msb of the CRC are sent first. Figure 29-1shows a typical HDLC frame.

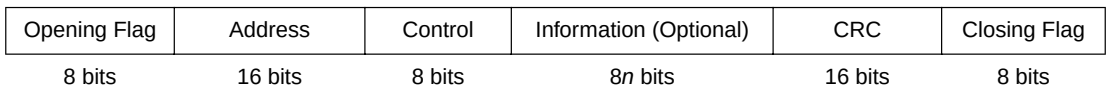


Figure 29-1. HDLC Framing Structure

After a closing flag is sent, the SCC updates the frame status bits of the BD and clears TxBD[R] (buffer ready). At the end of the current buffer, if TxBD[L] is not set (multiple buffers per frame), only TxBD[R] is cleared. Before the SCC proceeds to the next TxBD in the table, an interrupt can be issued if TxBD[I] is set. This interrupt programmability allows the core to intervene after each buffer, after a specific buffer, or after each frame.

The STOP TRANSMIT command can be used to expedite critical data ahead of previously linked buffers or to support efficient error handling. When the SCC receives a STOP

TRANSMIT command, it sends idles or flags instead of the current frame until it receives a RESTART TRANSMIT command. The GRACEFUL STOP TRANSMIT command can be used to insert a high-priority frame without aborting the current one—a graceful-stop-complete event is generated in SCCE[GRA] when the current frame is finished. See Section 29.6, “SCC HDLC Commands.”

29.3 SCC HDLC Channel Frame Reception

The HDLC receiver is designed to work with little or no core intervention to perform address recognition, CRC checking, and maximum frame length checking. Received frames can be used to implement any HDLC-based protocol.

Once enabled by the core, the receiver waits for an opening flag character. When it detects the first byte of the frame, the SCC compares the frame address with four user-programmable, 16-bit address registers and an address mask. The SCC compares the received address field with the user-defined values after masking with the address mask. To detect broadcast (all ones) address frames, one address register must be written with all ones.

If an address match is detected, the SCC fetches the next BD and SCC starts transferring the incoming frame to the buffer if it is empty. When the buffer is full, the SCC clears RxBD[E] and generates a maskable interrupt if RxBD[I] is set. If the incoming frame is larger than the current buffer, the SCC continues receiving using the next BD in the table.

During reception, the SCC checks for frames that are too long (using MFLR). When the frame ends, the CRC field is checked against the recalculated value and written to the buffer. RxBD[Data Length] of the last BD in the HDLC frame contains the entire frame length. This also enables software to identify the frames in which the maximum frame length violations occur. The SCC sets RxBD[L] (last buffer in frame), writes the frame status bits, and clears RxBD[E]. It then generates a maskable event (SCCE[RXF]) to indicate a frame was received. The SCC then waits for a new frame. Back-to-back frames can be received with only one shared flag between frames.

The received frames threshold parameter (RFTHR) can be used to postpone interrupts until a specified number of frames is received. This function can be combined with a timer to implement a timeout if fewer than the specified number of threshold frames is received.

Note that SCCs in HDLC mode, or any other synchronous mode, must receive a minimum of eight clocks after the last bit arrives to account for Rx FIFO delay.

29.4 SCC HDLC Parameter RAM

For HDLC mode, the protocol-specific area of the SCC parameter RAM is mapped as inTable 29-1.

Table 29-1. HDLC-Specific SCC Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x30	—	Word	Reserved
0x34	**C_MASK**	Word	CRC mask. For the 16-bit CRC-CCITT, initialize with 0x0000_F0B8. For 32-bit CRC-CCITT, initialize with 0xDEBB_20E3.
0x38	**C_PRES**	Word	CRC preset. For the 16-bit CRC-CCITT, initialize with 0x0000_FFFF. For 32-bit CRC-CCITT, initialize with 0xFFFF_FFFF.
0x3C	**DISFC**	Hword	Modulo 2^{16} counters maintained by the CP. Initialize them while the channel is disabled. DISFC (Discarded frame counter) Counts error-free frames discarded due to lack of free buffers. CRCEC (CRC error counter) Includes frames not addressed to the user or frames received in the BSY condition, but does not include overrun errors. ABTSC (Abort sequence counter) NMARC (Nonmatching address received counter) Includes error-free frames only. RETRC (Frame retransmission counter) Counts number of frames resent due to collision.
0x3E	**CRCEC**	Hword	
0x40	**ABTSC**	Hword	
0x42	**NMARC**	Hword	
0x44	**RETRC**	Hword	
0x46	**MFLR**	Hword	Max frame length register. The HDLC compares the incoming HDLC frame's length with the user-defined limit in MFLR. If the limit is exceeded, the rest of the frame is discarded and RxBD[LG] is set in the last BD of that frame. At the end of the frame the SCC reports frame status and frame length in the last RxBD. The MFLR is defined as all in-frame bytes between the opening and closing flags.
0x48	MAX_CNT	Hword	Maximum length counter. A temporary down-counter used to track frame length.
0x4A	**RFTHR**	Hword	Received frames threshold. Used to reduce potential interrupt overhead when each in a series of short HDLC frames causes an SCCE[RXF] event. Setting RFTHR determines the frequency of RXF interrupts, which occur only when the RFTHR limit is reached. Provide enough empty RxBDs for the number of frames specified in RFTHR.
0x4C	RFCNT	Hword	Received frames count. RFCNT is a down-counter used to implement RFTHR.
0x4E	**HMASK**	Hword	Mask register (HMASK) and four address registers (HADDR*n*) for address recognition. The SCC reads the frame address from the HDLC receiver, compares it with the HADDRs, and masks the result with HMASK. Setting an HMASK bit enables the corresponding comparison bit, clearing a bit masks it. When a match occurs, the frame address and data are written to the buffers. When no match occurs and a frame is error-free, the nonmatching address received counter (NMARC) is incremented. The eight low-order bits of HADDR*n* should contain the first address byte after the opening flag. For example, to recognize a frame that begins 0x7E (flag), 0x68, 0xAA, using 16-bit address recognition, HADDR*n* should contain 0xAA68 and HMASK should contain 0xFFFF. For 8-bit addresses, clear the eight high-order HMASK bits. See Figure 29-2.
0x50	**HADDR1**	Hword	
0x52	**HADDR2**	Hword	
0x54	**HADDR3**	Hword	
0x56	**HADDR4**	Hword	
0x58	TMP	Hword	Temporary storage.
0x5A	TMP_MB	Hword	Temporary storage.

[1] From SCC base. See Section 27.4.1, "SCC Base Addresses."

Figure 29-2 shows 16- and 8-bit address recognition.

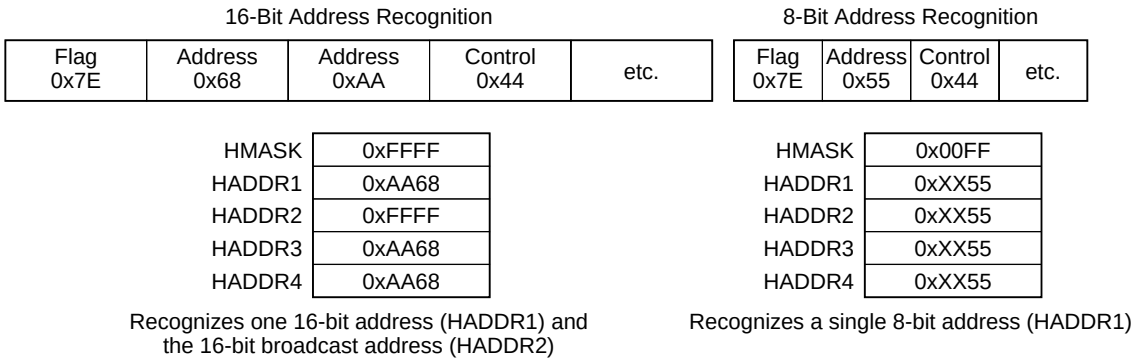


Figure 29-2. HDLC Address Recognition

29.5 Programming the SCC in HDLC Mode

HDLC mode is selected for an SCC by writing GSMR_L[MODE] = 0b0000. The HDLC controller uses the same buffer and BD data structure as other modes and supports multibuffer operation and address comparisons. Receive errors are reported through the RxBD; transmit errors are reported through the TxBD.

29.6 SCC HDLC Commands

The transmit and receive commands are issued to the CP command register (CPCR). Transmit commands are described in Table 29-2.

Table 29-2. Transmit Commands

Command	Description
STOP TRANSMIT	After a hardware or software reset and a channel is enabled in the GSMR, the transmitter starts polling the first BD in the TxBD table every 64 Tx clocks, or immediately if TODR[TOD] = 1, and begins sending data if TxBD[R] is set. If the SCC receives the STOP TRANSMIT command while not transmitting, the transmitter stops polling the BDs. If the SCC receives the command during transmission, transmission is aborted after a maximum of 64 additional bits, the Tx FIFO is flushed, and the current BD pointer TBPTR is not advanced (no new BD is accessed). The transmitter then sends an abort sequence (0x7F) and stops polling the BDs. When not transmitting, the channel sends flags or idles as programmed in the GSMR. Note that if PSMR[MFF] = 1, multiple small frames could be flushed from the Tx FIFO; a GRACEFUL STOP TRANSMIT command prevents this.
GRACEFUL STOP TRANSMIT	Stops transmission smoothly. Unlike a STOP TRANSMIT command, it stops transmission after the current frame is finished or immediately if no frame is being sent. SCCE[GRA] is set when transmission stops. HDLC Tx parameters and Tx BDs can then be updated. TBPTR points to the next TxBD. Transmission begins once TxBD[R] of the next BD is set and a RESTART TRANSMIT command is issued.

Table 29-2. Transmit Commands (continued)

Command	Description
RESTART TRANSMIT	Enables frames to be sent on the transmit channel. The HDLC controller expects this command after a STOP TRANSMIT is issued and the channel in its GSMR is disabled, after a GRACEFUL STOP TRANSMIT command, or after a transmitter error. The transmitter resumes from the current BD.
INIT TX PARAMETERS	Resets the Tx parameters in the parameter RAM. Issue only when the transmitter is disabled. INIT TX AND RX PARAMETERS resets both Tx and Rx parameters.

Receive commands are described in Table 29-3.

Table 29-3. Receive Commands

Command	Description
ENTER HUNT MODE	After a hardware or software reset, once an SCC is enabled in the GSMR, the receiver is automatically enabled and uses the first BD in the RxBD table. While the SCC is looking for the beginning of a frame, that SCC is in hunt mode. The ENTER HUNT MODE command is used to force the HDLC receiver to stop receiving the current frame and enter hunt mode, in which the HDLC continually scans the input data stream for a flag sequence. After receiving the command, the buffer is closed and the CRC is reset. Further frame reception uses the next BD.
CLOSE RXBD	Should not be used in the HDLC protocol.
INIT RX PARAMETERS	Resets the Rx parameters in the parameter RAM.; issue only when the receiver is disabled. Note that INIT TX AND RX PARAMETERS resets both Tx and Rx parameters.

29.7 Handling Errors in the SCC HDLC Controller

The SCC HDLC controller reports frame reception and transmission errors using BDs, error counters, and the SCCE. Transmission errors are described in Table 29-4.

Table 29-4. Transmit Errors

Error	Description
Transmitter Underrun	The channel stops transmitting, closes the buffer, sets TxBD[UN], and generates a TXE interrupt if not masked. Transmission resumes when a RESTART TRANSMIT command is issued. The SCC send and receive FIFOs are 32 bytes each.
$\overline{\text{CTS}}$ Lost during Frame Transmission	The channel stops transmitting, closes the buffer, sets TxBD[CT], and generates the TXE interrupt if not masked. Transmission resumes after a RESTART TRANSMIT command. If this error occurs on the first or second buffer of the frame and PSMR[RTE] = 1, the channel resends the frame when $\overline{\text{CTS}}$ is reasserted and no error is reported. If collisions are possible, to ensure proper retransmission of multi-buffer frames, the first two buffers of each frame should in total contain more than 36 bytes for SCC or 20 bytes for SCC. The channel also increments the retransmission counter RETRC in the parameter RAM.

Reception errors are described in Table 29-5.

Table 29-5. Receive Errors

Error	Description
Overrun	Each SCC maintains an internal FIFO for receiving data. The CP begins programming the SDMA channel (if the buffer is in external memory) and updating the CRC when a full or partial FIFO's worth of data (according to GSMR_H[RFW]) is received in the Rx FIFO. When an Rx FIFO overrun occurs, the previous byte is overwritten by the next byte. The previous data byte and the frame status are lost. The channel closes the buffer with RxBD[OV] set and generates an RXF interrupt if not masked. The receiver then enters hunt mode. Even if an overrun occurs during a frame whose address is not recognized, an RxBD with data length two is opened to report the overrun and the interrupt is generated.
$\overline{CD}$ Lost during Frame Reception	Highest priority error. The channel stops frame reception, closes the buffer, sets RxBD[CD], and generates the RXF interrupt if not masked. The rest of the frame is lost and other errors are not checked in that frame. At this point, the receiver enters hunt mode.
Abort Sequence	Occurs when seven or more consecutive ones are received. When this occurs while receiving a frame, the channel closes the buffer, sets RxBD[AB] and generates a maskable RXF interrupt. The channel also increments the abort sequence counter ABTSC. The CRC and nonoctet error status conditions are not checked on aborted frames. The receiver then enters hunt mode.
Nonoctet Aligned Frame	The channel writes the received data to the buffer, closes the buffer, sets RxBD[NO], and generates a maskable RXF interrupt. CRC error status should be disregarded on nonoctet frames. After a nonoctet aligned frame is received, the receiver enters hunt mode. An immediate back-to-back frame is still received. The nonoctet data may be derived from the last word in the buffer as follows: msb lsb [ ] [1] [0 0] Valid Data | Nonvalid Data Note that if buffer swapping is used (RFCR[BO] = 0b0x), the figure above refers to the last byte, rather than the last word, of the buffer. The lsb of each octet is sent first while the msb of the CRC is sent first.
CRC	The channel writes the received CRC to the buffer, closes the buffer, sets RxBD[CR], generates a maskable RXF interrupt, and increments the CRC error counter CRCEC. After receiving a frame with a CRC error, the receiver enters hunt mode. An immediate back-to-back frame is still received. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

29.8 HDLC Mode Register (PSMR)

The protocol-specific mode register (PSMR), shown in Figure 29-3, functions as the HDLC mode register.

	0–3	4–5	6	7	8	9	10	11	12	13–15
Field	NOF	CRC	RTE	—	FSE	DRT	BUS	BRM	MFF	—
Reset	0									
R/W	R/W									
Address	0x9_1A08 (PSMR1); 0x9_1A28 (PSMR2); 0x9_1A48 (PSMR3); 0x9_1A68 (PSMR4)									

Figure 29-3. HDLC Mode Register (PSMR)

Table 29-6 describes PSMR HDLC fields.

Table 29-6. PSMR HDLC Field Descriptions

Bits	Name	Description
0-3	NOF	Number of flags. Minimum number of flags between or before frames. If NOF = 0b0000, no flags are inserted between frames and the closing flag of one frame is followed by the opening flag of the next frame in the case of back-to-back frames. NOF can be modified on-the-fly.
4–5	CRC	CRC selection. 00 16-bit CCITT-CRC (HDLC). X16 + X12 + X5 + 1. x1 Reserved. 10 32-bit CCITT-CRC (HDLC). X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1.
6	RTE	Retransmit enable. 0 No retransmission. 1 Automatic frame retransmission is enabled. Particularly useful in the HDLC bus protocol and ISDN applications where multiple HDLC controllers can collide. Note that retransmission occurs only if a lost $\overline{\text{CTS}}$ occurs on the first or second buffer of the frame.
7	—	Reserved, should be cleared.
8	FSE	Flag sharing enable. FSE can be set only if GSMR_H[RTSM] is already set. Can be modified on-the-fly. 0 Normal operation. 1 If NOF[0–3] = 0b0000, a single shared flag is sent between back-to-back frames. Other values of NOF[0–3] are decremented by 1. Useful in signaling system #7 applications.
9	DRT	Disable receiver while transmitting. 0 Normal operation. 1 As the SCC sends data, the receiver is disabled and gated by the internal $\overline{\text{RTS}}$. This helps if the HDLC channel is on a multidrop line and the SCC does not need to receive its own transmission.
10	BUS	HDLC bus mode. 0 Normal HDLC operation. 1 HDLC bus operation is selected. See Section 29.14, “HDLC Bus Mode with Collision Detection.”
11	BRM	HDLC bus $\overline{\text{RTS}}$ mode. Valid only if BUS = 1. Otherwise, it is ignored. 0 Normal $\overline{\text{RTS}}$ operation during HDLC bus mode. $\overline{\text{RTS}}$ is asserted on the first bit of the Tx frame and negated after the first collision bit is received. 1 Special $\overline{\text{RTS}}$ operation during HDLC bus mode. $\overline{\text{RTS}}$ is delayed by one bit with respect to the normal case, which helps when the HDLC bus protocol is being run locally and sent over a long-distance line at the same time. The one-bit delay allows $\overline{\text{RTS}}$ to be used to enable the transmission line buffers so that the electrical effects of collisions are not sent over the transmission line.
12	MFF	Multiple frames in Tx FIFO. The receiver is not affected. 0 Normal operation. The Tx FIFO must never contain more than one HDLC frame. The $\overline{\text{CTS}}$ lost status is reported accurately on a per-frame basis. 1 The Tx FIFO can hold multiple frames, but lost $\overline{\text{CTS}}$ may not be reported on the buffer/frame it occurred on. This can improve performance of HDLC transmissions of small back-to-back frames or when the number of flags between frames should be limited.
13–15	—	Reserved, should be cleared.

29.9 SCC HDLC Receive Buffer Descriptor (RxBD)

The CP uses the RxBD, shown in Figure 29-4, to report on data received for each buffer.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	**E**	—	**W**	**I**	L	F	**CM**	—	DE	—	LG	NO	AB	CR	OV	CD
Offset + 2	Data Length															
Offset + 4	**Rx Buffer Pointer**															
Offset + 6																

Figure 29-4. SCC HDLC Receive Buffer Descriptor (RxBD)

Table 29-7describes HDLC RxBD status and control fields.

Table 29-7. SCC HDLC RxBD Status and Control Field Descriptions

Bits	Name	Description
0	**E**	Empty. 0 The buffer is full or reception stopped because of an error. The core can read or write to any fields of this RxBD. The CP does not use this BD while E = 0. 1 The buffer is not full. The CP controls the BD and buffer. The core should not update the BD.
1	—	Reserved, should be cleared.
2	**W**	Wrap (last BD in the RxBD table). 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CP receives incoming data using the BD pointed to by RBASE. The number of BDs in this table are programmable and determined only by RxBD[W] and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 SCCE[RXB] is not set after this buffer is used; SCCE[RXF] is unaffected. 1 SCCE[RXB] or SCCE[RXF] is set when the SCC uses this buffer.
4	L	Last buffer in frame. 0 Not the last buffer in frame. 1 Last buffer in frame. Indicates reception of a closing flag or an error, in which case one or more of the CD, OV, AB, and LG bits are set. The SCC writes the number of frame octets to the data length field.
5	F	First in frame. 0 Not the first buffer in a frame. 1 First buffer in a frame.
6	**CM**	Continuous mode. Note that RxBD[E] is cleared if an error occurs during reception, regardless of CM. 0 Normal operation. 1 RxBD[E] is not cleared by the CP after this BD is closed, allowing the associated buffer to be overwritten next time the CP accesses it.
7	—	Reserved, should be cleared.
8	DE	DPLL error. Set when a DPLL error occurs while this buffer is being received. DE is also set due to a missing transition when using decoding modes in which a transition is required for every bit. Note that when a DPLL error occurs, the frame closes and error checking halts.
9	—	Reserved, should be cleared.
10	LG	Rx frame length violation. Set when a frame larger than the maximum defined for this channel is recognized. Only the maximum-allowed number of bytes (MFLR) is written to the buffer. This event is not reported until the buffer is closed, SCCE[RXF] is set, and the closing flag is received. The total number of bytes received between flags is still written to the data length field.
11	NO	Rx nonoctet aligned frame. Set when a received frame contains a number of bits not divisible by eight.

Table 29-7. SCC HDLC RxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
12	AB	Rx abort sequence. Set when at least seven consecutive ones are received during frame reception.
13	CR	Rx CRC error. Set when a frame contains a CRC error. CRC bytes received are always written to the Rx buffer.
14	OV	Overrun. Set when a receiver overrun occurs during frame reception.
15	CD	Carrier detect lost (NMSI mode only). Set when $\overline{CD}$ is negated during frame reception.

Data length and buffer pointer fields are described in Section 27.3, “SCC Buffer Descriptors (BDs).” Because HDLC is a frame-based protocol, RxBD[Data Length] of the last buffer of a frame contains the total number of frame bytes, including the 2 or 4 bytes for CRC. Figure 29-5 shows an example of how RxBDs are used in receiving.

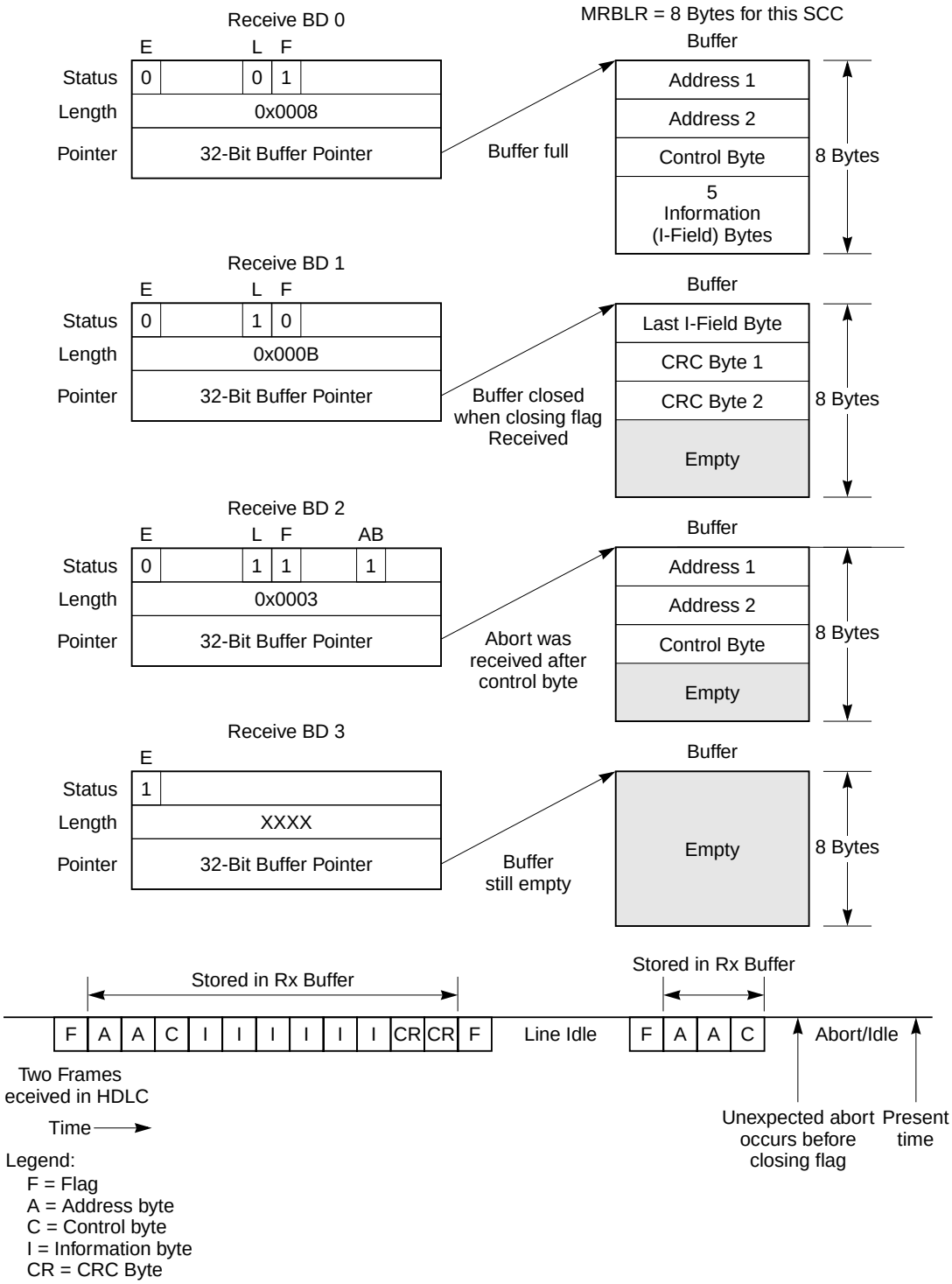


Figure 29-5. SCC HDLC Receiving Using RxBDs

29.10 SCC HDLC Transmit Buffer Descriptor (TxBD)

The CP uses the TxBD, shown in Figure 29-6, to confirm transmissions and indicate error conditions.

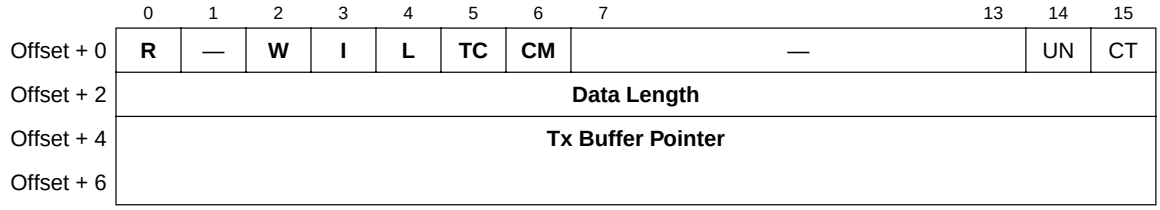


Figure 29-6. SCC HDLC Transmit Buffer Descriptor (TxBD)

Table 29-8 describes HDLC TxBD status and control fields.

Table 29-8. SCC HDLC TxBD Status and Control Field Descriptions

Bits	Name	Description
0	**R**	Ready. 0 The buffer is not ready for transmission. Both the buffer and the BD can be updated. The CP clears R after the buffer is sent or an error is encountered. 1 The buffer has not been sent or is being sent and the BD cannot be updated.
1	—	Reserved, should be cleared.
2	**W**	Wrap (last BD in TxBD table). 0 Not the last BD in the table. 1 Last BD in the BD table. After this buffer is used, the CP sends data using the BD pointed to by TBASE. The number of TxBDs in this table is determined by TxBD[W] and the space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is processed. 1 SCCE[TXB] or SCCE[TXE] is set when this buffer is processed, causing interrupts if not masked.
4	**L**	Last. 0 Not the last buffer in the frame. 1 Last buffer in the frame.
5	**TC**	Tx CRC. Valid only when TxBD[L] = 1. Otherwise, it is ignored. 0 Transmit the closing flag after the last data byte. This setting can be used to send a bad CRC after the data for testing purposes. 1 Transmit the CRC sequence after the last data byte.
6	**CM**	Continuous mode. 0 Normal operation. 1 The CP does not clear TxBD[R] after this BD is closed allowing the buffer to be resent the next time the CP accesses this BD. However, TxBD[R] is cleared if an error occurs during transmission, regardless of CM.
7–13	—	Reserved, should be cleared.
14	UN	Underrun. Set after the SCC sends a buffer and a transmitter underrun occurred.
15	CT	$\overline{\text{CTS}}$ lost. Indicates when $\overline{\text{CTS}}$ in NMSI mode or layer 1 grant is lost in GCI or IDL mode during frame transmission. If data from more than one buffer is currently in the FIFO when this error occurs, the HDLC writes CT in the current BD after sending the buffer.

The data length and buffer pointer fields are described in Section 27.3, “SCC Buffer Descriptors (BDs).”

29.11 HDLC Event Register (SCCE)/HDLC Mask Register (SCCM)

The SCC event register (SCCE) is used as the HDLC event register to report events recognized by the HDLC channel and to generate interrupts. When an event is recognized, the SCC sets the corresponding SCCE bit. Interrupts generated through SCCE can be masked in the SCC mask register (SCCM) which has the same bit format as the SCCE. Setting an SCCM bit enables the corresponding interrupt; clearing a bit masks it. SCCE bits are cleared by writing ones; writing zeros has no effect. All unmasked bits must be cleared before the CP clears the internal interrupt request. Figure 29-7 shows SCCE/SCCM for HDLC operation.

	0–2	3	4	5	6	7	8	9–10	11	12	13	14	15
Field	—	GLR	GLT	DCC	FLG	IDL	GRA	—	TXE	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000												
R/W	R/W												
Addr	0x9_1A10 (SCCE1); 0x9_1A30 (SCCE2); 0x9_1A50 (SCCE3); 0x9_1A70 (SCCE4) 0x9_1A14 (SCCM1); 0x9_1A34 (SCCM2); 0x9_1A54 (SCCM3); 0x9_1A74 (SCCM4)												

Figure 29-7. HDLC Event Register (SCCE)/HDLC Mask Register (SCCM)

Table 29-9 describes SCCE/SCCM fields.

Table 29-9. SCCE/SCCM Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared.
3, 4	GLR/GLT	Glitch on Rx/Tx. Set when the SCC detects a clock glitch on the receive/transmit clock. See Section 27.4.7, “Clock Glitch Detection.”
5	DCC	DPLL carrier sense changed. Set when the carrier sense status generated by the DPLL changes. Real-time status can be read in SCCS[CS]. This is not the $\overline{CD}$ status reported in port C. Valid only when the DPLL is used.
6	FLG	Flag status. Set when the SCC stops or starts receiving HDLC flags. Real-time status can be read in SCCS[FG].
7	IDL	Idle sequence status changed. Set when HDLC line status changes. Real-time status of the line can be read in SCCS[ID].
8	GRA	Graceful stop complete. A GRACEFUL STOP TRANSMIT command completed execution. Set as soon as the transmitter has sent a frame in progress when the command was issued. Set immediately if no frame was in progress when the command was issued.
9–10	—	Reserved, should be cleared.
11	TXE	Tx error. Indicates an error ($\overline{CTS}$ lost or underrun) has occurred on the transmitter channel.

Table 29-9. SCCE/SCCM Field Descriptions (continued)

Bits	Name	Description
12	RXF	Rx frame. Set when the number of receive frames specified in RFTHR are received on the HDLC channel. It is set no sooner than two clocks after the last bit of the closing flag is received. This event is not maskable via the RxBD[I] bit.
13	BSY	Busy condition. Indicates a frame arrived but was discarded due to a lack of buffers.
14	TXB	Transmit buffer. Enabled by setting TxBD[I]. TXB is set when a buffer is sent on the HDLC channel. For the last buffer in the frame, TXB is not set before the last bit of the closing flag begins its transmission; otherwise, it is set after the last byte of the buffer is written to the Tx FIFO.
15	RXB	Receive buffer. Enabled by setting RxBD[I]. RXB is set when the HDLC channel receives a buffer that is not the last in a frame.

Figure 29-8 shows interrupts that can be generated using the HDLC protocol.

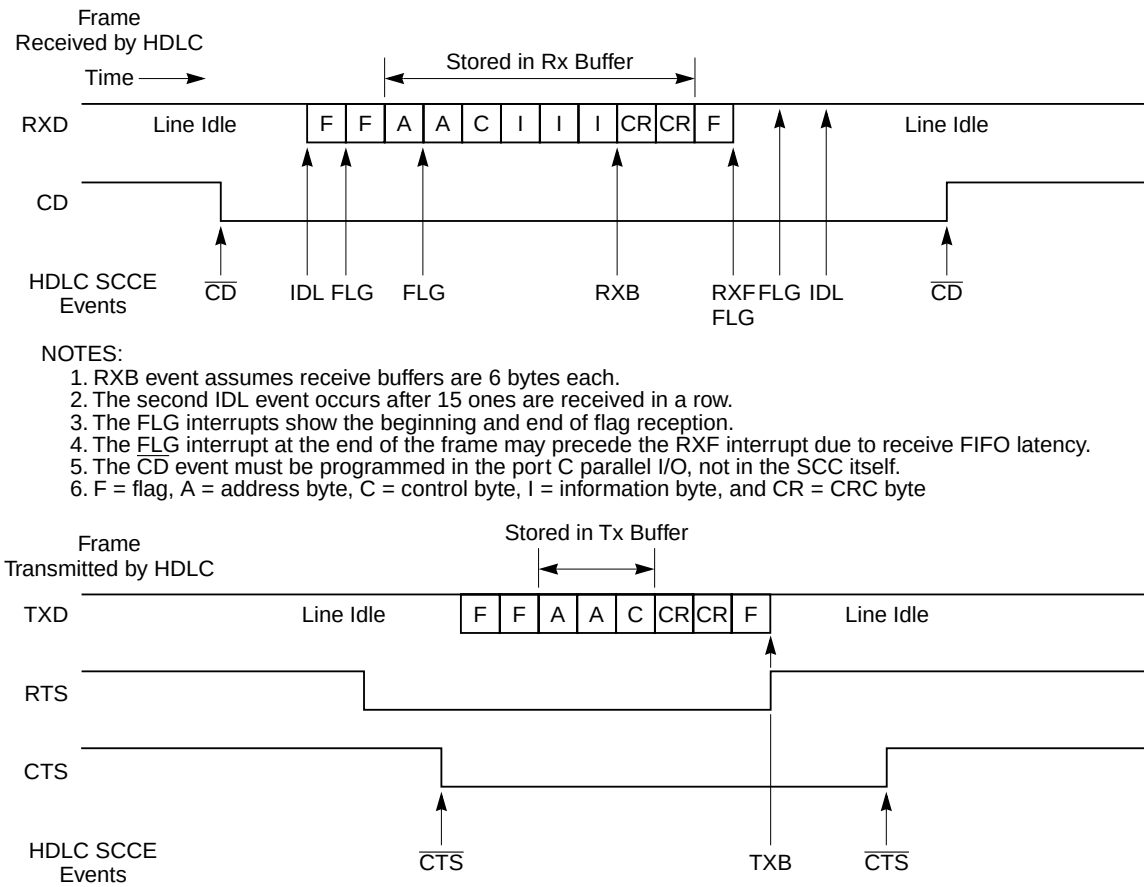


Figure 29-8. SCC HDLC Interrupt Event Example

29.12 SCC HDLC Status Register (SCCS)

The SCC status register (SCCS), shown in Figure 29-9, permits monitoring of real-time status conditions on RXD. The real-time status of $\overline{CTS}$ and $\overline{CD}$ are part of the port C parallel I/O.

	0 — 4	5	6	7
Field	—	FG	CS	ID
Reset	0000_0000			
R/W	R			
Addr	0x9_1A17 (SCCS1); 0x9_1A37 (SCCS2); 0x9_1A57 (SCCS3); 0x9_1A77 (SCCS4)			

Figure 29-9. SCC HDLC Status Register (SCCS)

Table 29-10 describes HDLC SCCS fields.

Table 29-10. HDLC SCCS Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	FG	Flags. The line is checked after the data has been decoded by the DPLL. 0 HDLC flags are not being received. The most recently received 8 bits are examined every bit time to see if a flag is present. 1 HDLC flags are being received. FG is set as soon as an HDLC flag (0x7E) is received on the line. Once it is set, it remains set at least 8 bit times and the next eight received bits are examined. If another flag occurs, FG stays set for at least another eight bits. If not, it is cleared and the search begins again.
6	CS	Carrier sense (DPLL). Shows the real-time carrier sense of the line as determined by the DPLL. 0 The DPLL does not sense a carrier. 1 The DPLL senses a carrier.
7	ID	Idle status. 0 The line is busy. 1 Set when RXD is a logic 1 (idle) for 15 or more consecutive bit times. It is cleared after a single logic 0 is received.

29.13 SCC HDLC Programming Examples

The following sections show examples for programming SCCs in HDLC mode. The first example uses an external clock. The second example implements Manchester encoding.

29.13.1SCC HDLC Programming Example #1

The following initialization sequence is for an SCC HDLC channel with an external clock. SCC2 is used with $\overline{\text{RTS2}}$, $\overline{\text{CTS2}}$, and $\overline{\text{CD2}}$ active; CLK3 is used for both the HDLC receiver and transmitter.

1. Configure port D pins to enable TXD2 and RXD2. Set PPARD[27,28] and PDIRD[27] and clear PDIRD[28] and PSORD[27,28].
2. Configure ports C and D pins to enable RTS2, CTS2 and CD2. Set PPARD[26], PPARC[12,13] and PDIRD[26] and clear PDIRC[12,13], PSORC[12,13] and PSORD[26].
3. Configure port C pin 29 to enable the CLK3 pin. Set PPARC[29] and clear PDIRC[29] and PSORC[29].
4. Connect CLK3 to SCC2 using the CPM mux. Write 0b110 to CMXSCR[R2CS] and CMXSCR[T2CS].
5. Connect the SCC2 to the NMSI (its own set of pins). clear CMXSCR[SC2].
6. Write RBASE and TBASE in the SCC2 parameter RAM to point to the RxBD and TxBD tables in dual-port RAM. Assuming one RxBD at the start of dual-port RAM and one TxBD following it, write RBASE with 0x0000 and TBASE with 0x0008.
7. Write RBASE and TBASE in the SCC2 parameter RAM to point to the RxBD and TxBD tables in dual-port RAM. Assuming one RxBD at the start of dual-port RAM and one TxBD following it, write RBASE with 0x0000 and TBASE with 0x0008.
8. Write 0x04A1_0000 to CPCR to execute the INIT RX AND TX PARAMS command for SCC2. This command updates RBPTR and TBPTR of the serial channel with the new values of RBASE and TBASE.
9. Write RFCR with 0x10 and TFCR with 0x10 for normal operation.
10. Write MRBLR with the maximum number of bytes per Rx buffer. Choose 256 bytes (MRBLR = 0x0100) so an entire Rx frame can fit in one buffer.
11. Write C_MASK with 0x0000F0B8 to comply with 16-bit CCITT-CRC.
12. Write C_PRES with 0x0000FFFF to comply with 16-bit CCITT-CRC.
13. Clear DISFC, CRCEC, ABTSC, NMARC, and RETRC for clarity.
14. Write MFLR with 0x0100 so the maximum frame size is 256 bytes.
15. Write RFTHR with 0x0001 to allow interrupts after each frame.
16. Write HMASK with 0x0000 to allow all addresses to be recognized.
17. Clear HADDR1–HADDR4 for clarity.
18. Initialize the RxBD. Assume the buffer is at 0x0000_1000 in main memory. RxBD[Status and Control]= 0xB000, RxBD[Data Length] = 0x0000 (not required), and RxBD[Buffer Pointer] = 0x0000_1000.

19. Initialize the TxBD. Assume the Tx data frame is at 0x0000_2000 in main memory and contains five 8-bit characters. TxBD[Status and Control] = 0xBC00, TxBD[Data Length] = 0x0005, and TxBD[Buffer Pointer] = 0x0000_2000.
20. Write 0xFFFF to SCCE to clear any previous events.
21. Write 0x001A to SCCM to enable TXE, RXF, and TXB interrupts.
22. Write 0x0040_0000 to the SIU interrupt mask register low (SIMR_L) so the SMC1 can generate a system interrupt. Initialize SIU interrupt pending register low (SIPNR_L) by writing 0xFFFF_FFFF to it.
23. Write 0x0000_0000 to GSMR_H2 to enable normal $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ behavior with idles (not flags) between frames.
24. Write 0x0000_0000 to GSMR_L2 to configure $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ to control transmission and reception in HDLC mode. Normal Tx clock operation is used. Notice that the transmitter (ENT) and receiver (ENR) have not been enabled. If inverted HDLC operation is preferred, set RINV and TINV.
25. Write 0x0000 to PSMR2 to configure one opening and one closing flag, 16-bit CCITT-CRC, and prevent multiple frames in the FIFO.
26. Write 0x00000030 to GSMR_L2 to enable the SCC2 transmitter and receiver. This additional write ensures that ENT and ENR are enabled last.

Note that after 5 bytes and CRC have been sent, the Tx buffer is closed; the Rx buffer is closed after a frame is received. Frames larger than 256 bytes cause a busy (out-of-buffers) condition because only one RxBD is prepared.

29.13.2SCC HDLC Programming Example #2

The following sequence initializes an HDLC channel that uses the DPLL in a Manchester encoding. Provide a clock which is 16× the chosen bit rate of CLK3. Then connect CLK3 to the HDLC transmitter and receiver. (A baud rate generator could be used instead.) Configure SCC2 to use $\overline{\text{RTS2}}$, $\overline{\text{CTS2}}$, and $\overline{\text{CD2}}$.

1. Follow steps 1–22 in example #1 above.
2. Write 0x004A_A400 to GSMR_L2 to make carrier sense always active, a 16-bit preamble of ‘01’ patterns, 16× operation of the DPLL and Manchester encoding for the receiver and transmitter, and HDLC mode. $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ should be configured to control transmission and reception. Do not set GSMR[ENT, ENR].
3. Write 0x0000 to PSMR2 to use one opening and one closing flag and 16-bit CCITT-CRC and to reject multiple frames in the FIFO.
4. Write 0x004A_A430 to GSMR_L2 to enable the SCC2 transmitter and receiver. This additional write to GSMR_L2 ensures that ENT and ENR are enabled last.

29.14 HDLC Bus Mode with Collision Detection

The HDLC controller includes an option for hardware collision detection and retransmission on an open-drain connected HDLC bus, referred to as HDLC bus mode. Most HDLC-based controllers provide only point-to-point communications; however, the HDLC bus enhancement allows implementation of an HDLC-based LAN and other point-to-multipoint configurations. The HDLC bus is based on techniques used in the CCITT ISDN I.430 and ANSI T1.605 standards for D-channel point-to-multipoint operation over the S/T interface. However, the HDLC bus does not fully comply with I.430 or T1.605 and cannot replace devices that implement these protocols. Instead, it is more suited to non-ISDN LAN and point-to-multipoint configurations.

Review the basic features of the I.430 and T1.605 before learning about the HDLC bus. The I.430 and T1.605 define a way to connect eight terminals over the D-channel of the S/T ISDN bus. The layer 2 protocol is a variant of HDLC, called LAPD. However, at layer 1, a method is provided to allow the eight terminals to send frames to the switch through the physical S/T bus.

To determine whether a channel is clear, the S/T interface device looks at an echo bit on the line designed to echo the last bit sent on the D channel. Depending on the class of terminal and the context, an S/T interface device waits for 7–10 ones on the echo bit before letting the LAPD frame begin transmission, after which the S/T interface monitors transmitted data. As long as the echo bit matches the sent data, transmission continues. If the echo bit is ever 0 when the transmit bit is 1, a collision occurs between terminals; the station(s) that sent a zero stops transmitting. The station that sent a 1 continues as normal.

The I.430 and T1.605 standards provide a physical layer protocol that allows multiple terminals to share one physical connection. These protocols handle collisions efficiently because one station can always complete its transmission, at which point, it lowers its own priority to give other devices fair access to the physical connection.

The HDLC bus differs from the I.430 and T1.605 standards as follows:

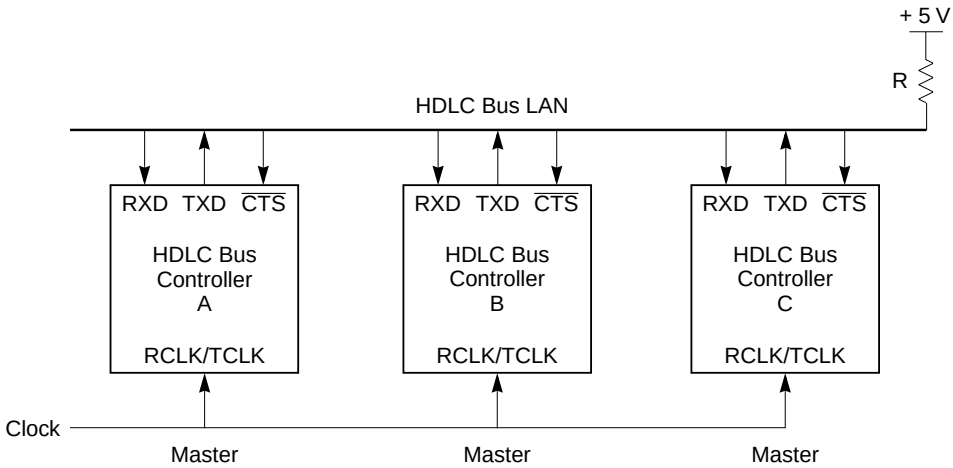
- The HDLC bus uses a separate input signal rather than the echo bit to monitor data; the transmitted data is simply connected to the $\overline{\text{CTS}}$ input.
- The HDLC bus is a synchronous, digital open-drain connection for short-distance configurations, rather than the more complex S/T interface.
- Any HDLC-based frame protocol can be used at layer 2, not just LAPD.
- HDLC bus devices wait 8–10 rather than 7–10 bit times before transmitting. (HDLC bus has only one class.)

The collision-detection mechanism supports only:

- NRZ-encoded data
- A common synchronous clock for all receivers and transmitters

- Non-inverted data (GSMR[RINV, TINV] = 0)
- Open-drain connection with no external transceivers

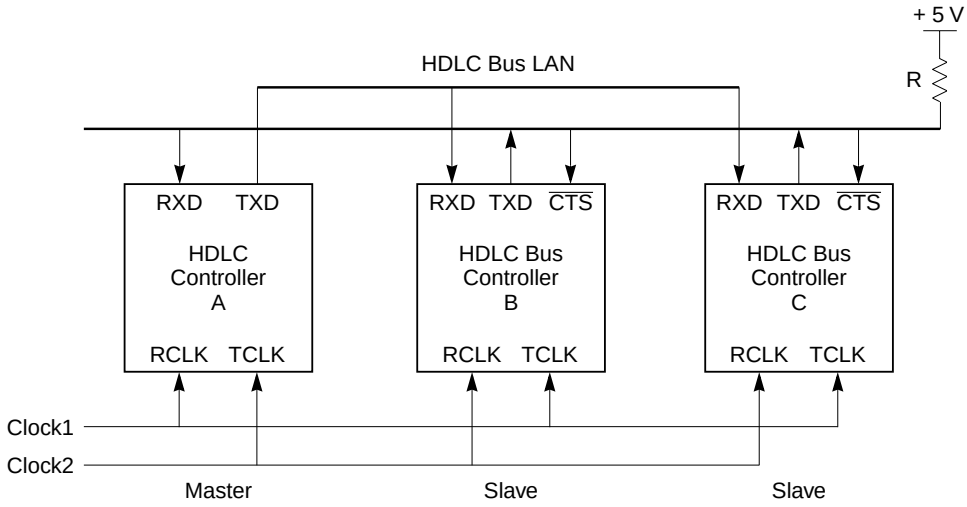
Figure 29-10 shows the most common HDLC bus LAN configuration, a multimaster configuration. A station can transfer data to or from any other LAN station. Transmissions are half-duplex, which is typical in LANs.



NOTES:
1. Transceivers may be used to extend the LAN size.
2. The TXD pins of slave devices should be configured to open-drain in the port C parallel I/O port.
3. Clock is a common RCLK/TCLK for all stations.

Figure 29-10. Typical HDLC Bus Multimaster Configuration

In single-master configuration, a master station transmits to any slave station without collisions. Slaves communicate only with the master, but can experience collisions in their access over the bus. In this configuration, a slave that communicates with another slave must first transmit its data to the master, where the data is buffered in RAM and then resent to the other slave. The benefit of this configuration, however, is that full-duplex operation can be obtained. In a point-to-multipoint environment, this is the preferred configuration. Figure 29-11 shows the single-master configuration.



NOTES:
1. Transceivers may be used to extend the LAN size.
2. The TXD pins of slave devices should be configured to open-drain in the port C parallel I/O port.
3. Clock1 is the master RCLK and the slave TCLK.
4. Clock2 is the master TCLK and the slave RCLK.

Figure 29-11. Typical HDLC Bus Single-Master Configuration

29.14.1 HDLC Bus Features

The main features of the HDLC bus are as follows:

- Superset of the HDLC controller features
- Automatic HDLC bus access
- Automatic retransmission in case of collision
- May be used with the NMSI or a TDM bus
- Delayed $\overline{\text{RTS}}$ mode

29.14.2 Accessing the HDLC Bus

The HDLC bus protocol ensures orderly bus control when multiple transmitters attempt simultaneous access. The transmitter sending a zero bit at the time of collision completes the transmission. If a station sends out an opening flag (0x7E) while another station is already sending, the collision is always detected within the first byte, because the transmission in progress is using zero bit insertion to prevent flag imitation.

While in the active condition (ready to transmit), the HDLC bus controller monitors the bus using $\overline{\text{CTS}}$. It counts the one bits on $\overline{\text{CTS}}$. When eight consecutive ones are counted, the HDLC bus controller starts transmitting on the line; if a zero is detected, the internal counter is cleared. During transmission, data is continuously compared with the external bus using $\overline{\text{CTS}}$. $\overline{\text{CTS}}$ is sampled halfway through the bit time using the rising edge of the

Tx clock. If the transmitted bit matches the received $\overline{CTS}$ bus sample, transmission continues. However, if the received $\overline{CTS}$ sample is 0 and the transmitted bit is 1, transmission stops after that bit and waits for an idle line before attempting retransmission. Since the HDLC bus uses a wired-OR scheme, a transmitted zero has priority over a transmitted 1. Figure 29-12 shows how $\overline{CTS}$ is used to detect collisions.

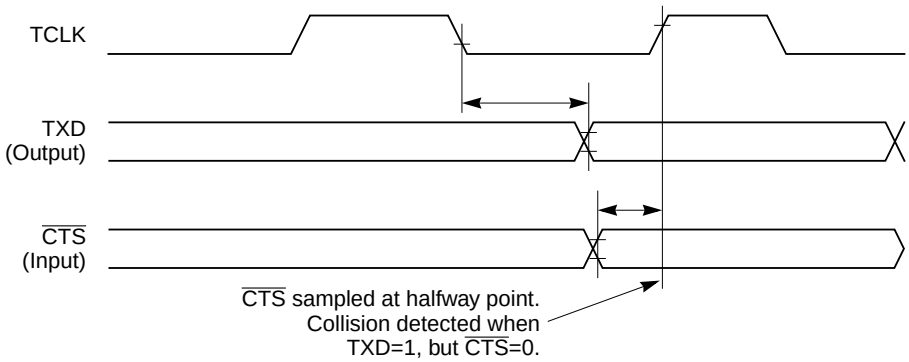


Figure 29-12. Detecting an HDLC Bus Collision

If both the destination address and source address are included in the HDLC frame, then a predefined priority of stations results; if two stations begin to transmit simultaneously, they necessarily detect a collision no later than the end of the source address.

The HDLC bus priority mechanism ensures that stations share the bus equally. To minimize idle time between messages, a station normally waits for eight one bits on the line before attempting transmission. After successfully sending a frame, a station waits for 10 rather than eight consecutive one bits before attempting another transmission. This mechanism ensures that another station waiting to transmit acquires the bus before a station can transmit twice. When a low priority station detects 10 consecutive ones, it tries to transmit; if it fails, it reinstates the high priority of waiting for only eight ones.

29.14.3Increasing Performance

Because it uses a wired-OR configuration, HDLC bus performance is limited by the rise time of the one bit. To increase performance, give the one bit more rise time by using a clock that is low longer than it is high, as shown in Figure 29-13.

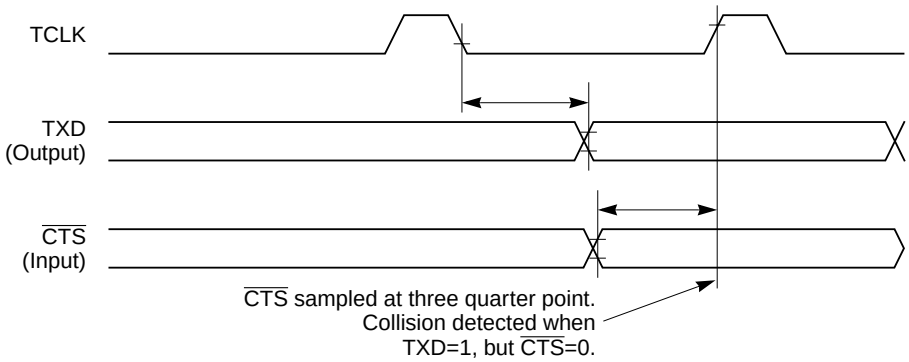
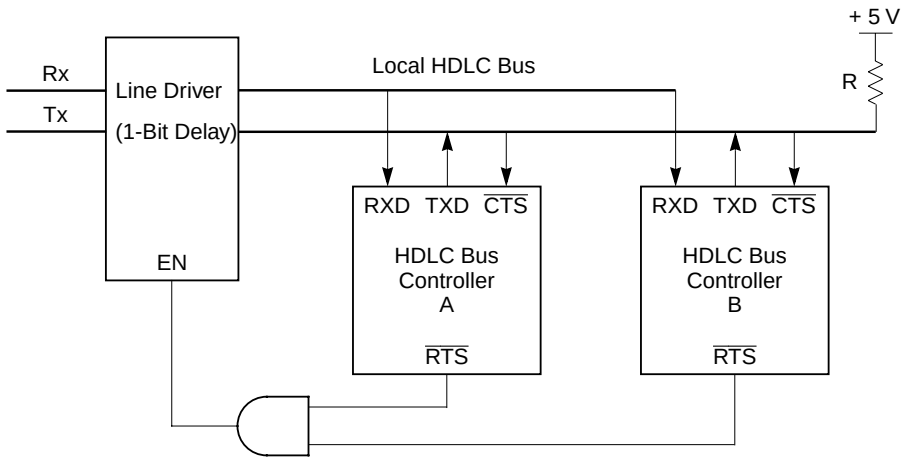


Figure 29-13. Nonsymmetrical Tx Clock Duty Cycle for Increased Performance

29.14.4 Delayed RTS Mode

Figure 29-14 shows local HDLC bus controllers using a standard transmission line and a local bus. The controllers do not communicate with each other but with a station on the transmission line; yet the HDLC bus protocol controls access to the transmission line.



NOTES:
1. The TXD pins of slave devices should be configured to open-drain in the port C parallel I/O port.
2. The RTS pins of each HDLC bus controller are configured to delayed RTS mode.

Figure 29-14. HDLC Bus Transmission Line Configuration

Normally, RTS goes active at the beginning of the opening flag's first bit. Setting PSMR[BRM] delays RTS by one bit, which is useful when the HDLC bus connects multiple local stations to a transmission line. If the transmission line driver has a one-bit delay, the delayed RTS can be used to enable the output of the line driver. As a result, the electrical effects of collisions are isolated locally. Figure 29-15 shows RTS timing.

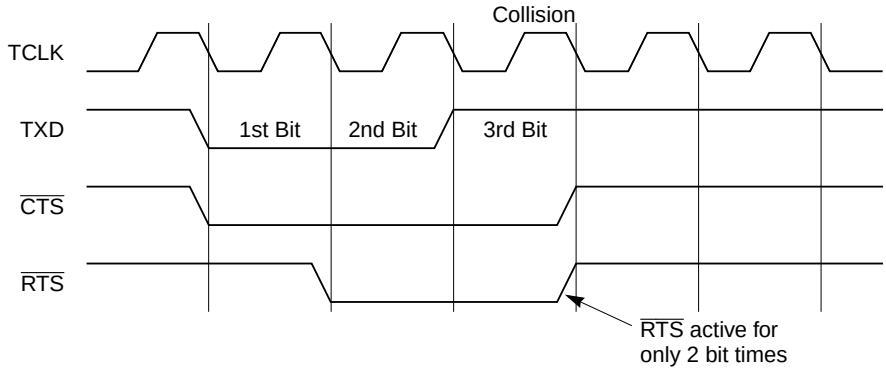
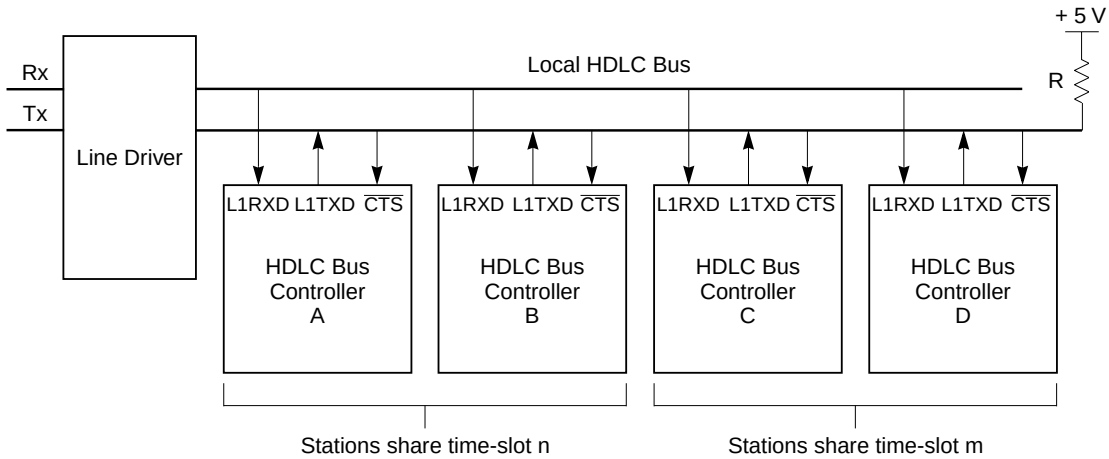


Figure 29-15. Delayed RTS Mode

29.14.5Using the Time-Slot Assigner (TSA)

HDLC bus controllers can be used with a time-division multiplexed transmission line and a local bus, as shown in Figure 29-16. Local stations use time slots to communicate over the TDM transmission line; stations that share a time slot use the HDLC bus protocol to control access to the local bus.



NOTES:
1. All TXD pins of slave devices should be configured to open-drain in the port C parallel I/O port.
2. The TSA in the SI of each station is used to configure the preferred time slot.
3. The choice of the number of stations to share a time slot is user-defined. It is two in this example.

Figure 29-16. HDLC Bus TDM Transmission Line Configuration

The local SCCs in HDLC bus mode communicate only with the transmission line and not with each other. The SCCs use the TSA of the serial interface, receiving and sending data over L1TXD*x* and L1RXD*x*. Because collisions are still detected from the individual SCC CTS pin, it must be configured to connect to the chosen SCC. Because the SCC only receives clocks during its time slot, CTS is sampled only during the Tx clock edges of the particular SCC time slot.

29.14.6 HDLC Bus Protocol Programming

The HDLC bus on the MPC8560 is implemented using the SCC in HDLC mode with bus-specific options selected in the PSMR and GSMR, as outlined below. See also Section 29.5, “Programming the SCC in HDLC Mode.”

29.14.6.1 Programming GSMR and PSMR for the HDLC Bus Protocol

To program the protocol-specific mode register (PSMR), set the bits as described below:

- Configure NOF as preferred
- Set RTE and BUS to 1
- Set BRM to 1 if delayed $\overline{\text{RTS}}$ is desired
- Configure CRC to 16-bit CRC CCITT (0b00).
- Configure other bits to zero or default.

To program the general SCC mode register (GSMR), set the bits as described below:

- Set MODE to HDLC mode (0b0000).
- Configure CTSS to 1 and all other bits to zero or default.
- Configure the DIAG bits for normal operation (0b00).
- Configure RDCR and TDCR for 1× clock (0b00).
- Configure TENC and RENC for NRZ (0b000).
- Clear RTSM to send idles between frames.
- Set GSMR_L[ENT, ENR] as the last step to begin operation.

29.14.6.2 HDLC Bus Controller Programming Example

Except for the above discussion in Section 29.14.6.1, “Programming GSMR and PSMR for the HDLC Bus Protocol,” use the example in Section 29.13.1, “SCC HDLC Programming Example #1.”

Chapter 30
SCC BISYNC Mode

The byte-oriented BISYNC protocol was developed by IBM for use in networking products. There are three classes of BISYNC frames—transparent, nontransparent with header, and nontransparent without header, shown in Figure 30-1. The transparent frame type in BISYNC is not related to transparent mode, discussed in Chapter 31, “SCC Transparent Mode.” Transparent BISYNC mode allows full binary data to be sent with any possible character pattern. Each class of frame starts with a standard two-octet synchronization pattern and ends with a block check code (BCC). The end-of-text character (ETX) is used to separate the text and BCC fields.

Nontransparent with Header

SYN1	SYN2	SOH	Header	STX	Text	ETX	BCC

Nontransparent without Header

SYN1	SYN2	STX	Text	ETX	BCC

Transparent

SYN1	SYN2	DLE	STX	Transparent Text	DLE	ETX	BCC

Figure 30-1. Classes of BISYNC Frames

The bulk of a frame is divided into fields whose meaning depends on the frame type. The BCC is a 16-bit CRC format if 8-bit characters are used; it is a combination longitudinal (sum check) and vertical (parity) redundancy check if 7-bit characters are used. In transparent operation, a special character (DLE) is defined that tells the receiver that the next character is text, allowing BISYNC control characters to be valid text data in a frame. A DLE sent as data must be preceded by a DLE character. This is sometimes called byte-stuffing. The physical layer of the BISYNC communications link must synchronize the receiver and transmitter, usually by sending at least one pair of synchronization characters before each frame.

BISYNC protocol is unusual in that a transmit underrun need not be an error. If an underrun occurs, a synchronization pattern is sent until data is again ready. In nontransparent operation, the receiver discards additional synchronization characters (SYNCs) as they are

received. In transparent mode, DLE-SYNC pairs are discarded. Normally, for proper transmission, an underrun must not occur between the DLE and its following character. This failure mode cannot occur with the MPC8560.

An SCC can be configured as a BISYNC controller to handle basic BISYNC protocol in normal and transparent modes. The controller can work with the time-slot assigner (TSA) or nonmultiplexed serial interface (NMSI). The controller has separate transmit and receive sections whose operations are asynchronous with the core and either synchronous or asynchronous with other SCCs.

30.1 Features

The following list summarizes features of the SCC in BISYNC mode:

- Flexible data buffers
- Eight control character recognition registers
- Automatic SYNC1–SYNC2 detection
- 16-bit pattern (bisync)
- 8-bit pattern (monosync)
- 4-bit pattern (nibblesync)
- External SYNC pin support
- SYNC/DLE stripping and insertion
- CRC16 and LRC (sum check) generation/checking
- VRC (parity) generation/checking
- Supports BISYNC transparent operation
- Maintains parity error counter
- Reverse data mode capability

30.2 SCC BISYNC Channel Frame Transmission

The BISYNC transmitter is designed to work with almost no core intervention. When the transmitter is enabled, it starts sending SYN1–SYN2 pairs in the data synchronization register (DSR) or idles as programmed in the PSMR. The BISYNC controller polls the first BD in the channel's TxBD table. If there is a message to send, the controller fetches the message from memory and starts sending it after the SYN1–SYN2 pair. The entire pair is always sent, regardless of GSMR[SYNL].

After a buffer is sent, if the last (TxBD[L]) and the Tx block check sequence (TxBD[TB]) bits are set, the BISYNC controller appends the CRC16/LRC and then writes the message status bits in TxBD status and control fields and clears the ready bit, TxBD[R]. It then starts sending the SYN1–SYN2 pairs or idles, according to GSMR[RTSM]. If the end of the

current BD is reached and TxBD[L] is not set, only TxBD[R] is cleared. In both cases, an interrupt is issued according to TxBD[I]. TxBD[I] controls whether interrupts are generated after transmission of each buffer, a specific buffer, or each block. The controller then proceeds to the next BD.

If no additional buffers have been sent to the controller for transmission, an in-frame underrun is detected and the controller starts sending syncs or idles. If the controller is in transparent mode, it sends DLE-sync pairs. Characters are included in the block check sequence (BCS) calculation on a per-buffer basis. Each buffer can be programmed independently to be included or excluded from the BCS calculation; thus, excluded characters must reside in a separate buffer. The controller can reset the BCS generator before sending a specific buffer. In transparent mode, the controller inserts a DLE before sending a DLE character, so that only one DLE is used in the calculation.

30.3 SCC BISYNC Channel Frame Reception

Although the receiver is designed to work with almost no core intervention, the user can intervene on a per-byte basis if necessary. The receiver performs CRC16, longitudinal (LRC) or vertical redundancy (VRC) checking, sync stripping in normal mode, DLE-sync stripping, stripping of the first DLE in DLE-DLE pairs in transparent mode, and control character recognition. Control characters are discussed in Section 30.6, “SCC BISYNC Control Character Recognition.”

When enabled, the receiver enters hunt mode where the data is shifted into the receiver shift register one bit at a time and the contents of the shift register are compared to the contents of DSR[SYN1, SYN2]. If the two are unequal, the next bit is shifted in and the comparison is repeated. When registers match, hunt mode is terminated and character assembly begins. The controller is character-synchronized and performs SYNC stripping and message reception. It reverts to hunt mode when it receives an ENTER HUNT MODE command, an error condition, or an appropriate control character.

When receiving data, the controller updates the BCS bit in the BD for each byte transferred. When the buffer is full, the controller clears the E bit in the BD and generates an interrupt if the I bit in the BD is set. If incoming data exceeds the buffer length, the controller fetches the next BD; if E is zero, reception continues to its buffer.

When a BCS is received, it is checked and written to the buffer. The BISYNC controller sets the last bit, writes the message status bits into the BD, clears the E bit, and then generates a maskable interrupt, indicating that a block of data was received and is in memory. The BCS calculations do not include SYNCs (in nontransparent mode) or DLE-SYNC pairs (in transparent mode).

Note that GSMR_H[RFW] should be set for an 8-bit-wide receive FIFO for the BISYNC receiver. See Section 27.2, “General SCC Mode Registers (GSMR1–GSMR4).”

30.4 SCC BISYNC Parameter RAM

For BISYNC mode, the protocol-specific area of the SCC parameter RAM is mapped as in Table 30-1.

Table 30-1. SCC BISYNC Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x30	—	Word	Reserved
0x34	CRCC	Word	CRC constant temp value.
0x38	PRCRC	Hword	Preset receiver/transmitter CRC16/LRC. These values should be preset to all ones or zeros, depending on the BCS used.
0x3A	PTCRC	Hword	
0x3C	PAREC	Hword	Receive parity error counter. This 16-bit (modulo 2^{16}) counter maintained by the CP counts parity errors on receive if the parity feature of BISYNC is enabled. Initialize PAREC while the channel is disabled.
0x3E	BSYNC	Hword	BISYNC SYNC register. Contains the value of the SYNC to be sent as the second byte of a DLE–SYNC pair in an underrun condition and stripped from incoming data on receive once the receiver synchronizes to the data using the DSR and SYN1–SYN2 pair. See Section 30.7, “BISYNC SYNC Register (BSYNC).”
0x40	BDLE	Hword	BISYNC DLE register. Contains the value to be sent as the first byte of a DLE–SYNC pair and stripped on receive. See Section 30.8, “SCC BISYNC DLE Register (BDLE).”
0x42	CHARACTER1	Hword	Control character 1–8. These values represent control characters that the BISYNC controller recognizes. See Section 30.6, “SCC BISYNC Control Character Recognition.”
0x44	CHARACTER2	Hword	
0x46	CHARACTER3	Hword	
0x48	CHARACTER4	Hword	
0x4A	CHARACTER5	Hword	
0x4C	CHARACTER6	Hword	
0x4E	CHARACTER7	Hword	
0x50	CHARACTER8	Hword	
0x52	RCCM	Hword	Receive control character mask. Masks CHARACTER*n* comparison so control character classes can be defined. Setting a bit enables and clearing a bit masks comparison. See Section 30.6, “SCC BISYNC Control Character Recognition.”

[1] From SCC*x* base address. See Section 27.4.1, “SCC Base Addresses.”

GSMR[MODE] determines the protocol for each SCC. The SYN1–SYN2 synchronization characters are programmed in the DSR (see Section 27.2.2, “Data Synchronization Register (DSR).”) The BISYNC controller uses the same basic data structure as other modes; receive and transmit errors are reported through their respective BDs. There are two basic ways to handle BISYNC channels:

- The controller can inspect data on a per-byte basis and interrupt the core each time a byte is received.

- The controller can be programmed so software handles the first two or three bytes. The controller directly handles subsequent data without interrupting the core.

30.5 SCC BISYNC Commands

Transmit and receive commands are issued to the CP command register (CPCR). Transmit commands are described in Table 30-2.

Table 30-2. Transmit Commands

Command	Description
STOP TRANSMIT	After hardware or software is reset and the channel is enabled in the GSMR, the channel is in transmit enable mode and starts polling the first BD every 64 transmit clocks. This command stops transmission after a maximum of 64 additional bits without waiting for the end of the buffer and the transmit FIFO to be flushed. TBPTR is not advanced, no new BD is accessed, and no new buffers are sent for this channel. SYNC–SYNC or DLE–SYNC pairs are sent continually until a RESTART TRANSMIT is issued. A STOP TRANSMIT can be used when an EOT sequence should be sent and transmission should stop. After transmission resumes, the EOT sequence should be the first buffer sent to the controller. Note that the controller remains in transparent or normal mode after it receives a STOP TRANSMIT or RESTART TRANSMIT command.
GRACEFUL STOP TRANSMIT	Stops transmission after the current frame finishes sending or immediately if there is no frame being sent. SCCE[GRA] is set once transmission stops. Then BISYNC transmit parameters and TxBDs can be modified. The TBPTR points to the next TxBD. Transmission resumes when the R bit of the next BD is set and a RESTART TRANSMIT is issued.
RESTART TRANSMIT	Lets characters be sent on the transmit channel. The BISYNC controller expects it after a STOP TRANSMIT or a GRACEFUL STOP TRANSMIT command is issued, after a transmitter error occurs, or after a STOP TRANSMIT is issued and the channel is disabled in its SCCM. The controller resumes transmission from the current TBPTR in the channel's TxBD table.
INIT TX PARAMETERS	Initializes all transmit parameters in the serial channel's parameter RAM to their reset state. Issue only when the transmitter is disabled. INIT TX AND RX PARAMETERS resets transmit and receive parameters.

Receive commands are described in Table 30-3.

Table 30-3. Receive Commands

Command	Description
RESET BCS CALCULATION	Immediately resets the receive BCS accumulator. It can be used to reset the BCS after recognizing a control character, thus signifying that a new block is beginning.
ENTER HUNT MODE	After hardware or software is reset and the channel is enabled in SCCM, the channel is in receive enable mode and uses the first BD. This command forces the controller to stop receiving and enter hunt mode, during which the controller continually scans the data stream for an SYN1–SYN2 sequence as programmed in the DSR. After receiving the command, the current receive buffer is closed and the BCS is reset. Message reception continues using the next BD.
CLOSE RXBD	Used to force the SCC to close the current RxBD if it is in use and to use the next BD for subsequent data. If data is not being received, no action is taken.
INIT RX PARAMETERS	Initializes receive parameters in this serial channel's parameter RAM to reset state. Issue only when the receiver is disabled. An INIT TX AND RX PARAMETERS resets transmit and receive parameters.

30.6 SCC BISYNC Control Character Recognition

The BISYNC controller recognizes special control characters that customize the protocol implemented by the BISYNC controller and aid its operation in a DMA-oriented environment. They are used for receive buffers longer than one byte. In single-byte buffers, each byte can be easily inspected so control character recognition should be disabled.

The control character table, shown in Figure 30-2, lets the BISYNC controller recognize the end of the current block. Because the controller imposes no restrictions on the format of BISYNC blocks, software must respond to received characters and inform the controller of mode changes and of certain protocol events, such as resetting the BCS. Using the control character table correctly allows the remainder of the block to be received without interrupting software.

Up to eight control characters can be defined to inform the BISYNC controller that the end of the current block is reached and whether a BCS is expected after the character. For example, the end-of-text character (ETX) implies an end-of-block (ETB) with a subsequent BCS. An enquiry (ENQ) character designates an end of block without a subsequent BCS. All the control characters are written into the data buffer. The BISYNC controller uses a table of 16-bit entries to support control character recognition. Each entry consists of the control character, an end-of-table bit (E), a BCS expected bit (B), and a hunt mode bit (H). The RCCM entry defines classes of control characters that support masking option.

Offset from SCC*x* Base	0	1	2	3 — 7	8 — 15
0x42	E	B	H	—	CHARACTER1
0x44	E	B	H	—	CHARACTER2
0x46	E	B	H	—	CHARACTER3
0x48	E	B	H	—	CHARACTER4
0x4A	E	B	H	—	CHARACTER5
0x4D	E	B	H	—	CHARACTER6
0x4E	E	B	H	—	CHARACTER7
0x50	E	B	H	—	CHARACTER8
0x52	1	1	1	—	MASK VALUE(RCCM)

Figure 30-2. Control Character Table and RCCM

Table 30-4 describes control character table and RCCM fields.

Table 30-4. Control Character Table and RCCM Field Descriptions

Offset	Bits	Name	Description
0x42–0x50	0	E	End of table. 0 This entry is valid. The lower eight bits are checked against the incoming character. In tables with eight control characters, E should be zero in all eight positions. 1 The entry is not valid. No other valid entries exist beyond this entry.
	1	B	BCS expected. A maskable interrupt is generated after the buffer is closed. 0 The character is written into the receive buffer and the buffer is closed. 1 The character is written into the receive buffer. The receiver waits for one LRC or two CRC bytes of BCS and then closes the buffer. This should be used for ETB, ETX, and ITB.
	2	H	Hunt mode. Enables hunt mode when the current buffer is closed. 0 The BISYNC controller maintains character synchronization after closing this buffer. 1 The BISYNC controller enters hunt mode after closing the buffer. When the B bit is set, the controller enters hunt mode after receiving the BCS.
	3–7	—	Reserved
	8–15	CHARACTER*n*	Control character 1–8. When using 7-bit characters with parity, include the parity bit in the character value.
0x52	0–2	—	All ones.
	3–7	—	Reserved
	8–15	RCCM	Received control character mask. Masks comparison of CHARACTER*n*. Each bit of RCCM masks the corresponding bit of CHARACTER*n*. 0 Mask this bit in the comparison of the incoming character and CHARACTER*n*. 1 The address comparison on this bit proceeds normally and no masking occurs. If RCCM is not set, erratic operation can occur during control character recognition.

30.7 BISYNC SYNC Register (BSYNC)

The BSYNC register, as seen in Figure 30-3, defines BISYNC stripping and SYNC character insertion. When an underrun occurs, the BISYNC controller inserts SYNC characters until the next buffer is available for transmission. If the receiver is not in hunt mode when a SYNC character is received, it discards this character if the valid bit (BSYNC[V]) is set. When using 7-bit characters with parity, the parity bit should be included in the SYNC register value.

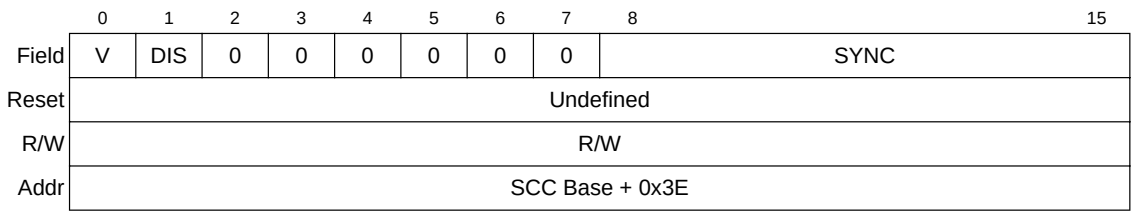


Figure 30-3. BISYNC SYNC (BSYNC)

Table 30-5 describes BSYNC fields.

Table 30-5. BSYNC Field Descriptions

Bits	Name	Description
0	V	Valid. If V = 1 and the receiver is not in hunt mode when a SYNC character is received, this character is discarded.
1	DIS	Disable BSYNC stripping 0 Normal mode. 1 BSYNC stripping disabled (BISYNC transparent mode only).
2–7	—	All zeros
8–15	SYNC	SYNC character

30.8 SCC BISYNC DLE Register (BDLE)

The BDLE register, as seen in Figure 30-4, is used to define the BISYNC stripping and insertion of DLE characters. When an underrun occurs while a message is being sent in transparent mode, the BISYNC controller inserts DLE-SYNC pairs until the next buffer is available for transmission.

In transparent mode, the receiver discards any DLE character received and excludes it from the BCS if the valid bit (BDLE[V]) is set. If the second character is SYNC, the controller discards it and excludes it from the BCS. If it is a DLE, the controller writes it to the buffer and includes it in the BCS. If it is not a DLE or SYNC, the controller examines the control character table and acts accordingly. If the character is not in the table, the buffer is closed with the DLE follow character error bit set. If the valid bit is not set, the receiver treats the character as a normal character. When using 7-bit characters with parity, the parity bit should be included in the DLE register value.

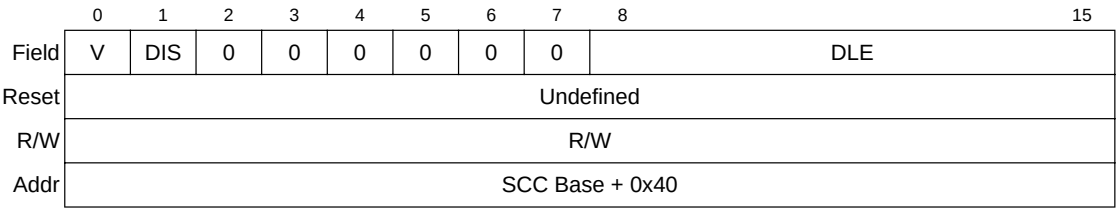


Figure 30-4. BISYNC DLE (BDLE)

Table 30-6 describes BDLE fields.

Table 30-6. BDLE Field Descriptions

Bits	Name	Description
0	V	Valid. If V = 1 and the receiver is not in hunt mode when a SYNC character is received, this character is discarded.
1	DIS	Disable DLE stripping 0 Normal mode. 1 DLE stripping disabled. When DIS is enabled in BDLE and on BSYNC the following cases occur: DLE-DLE sequence. Both characters are written to the memory. The BCS is calculated only on the second DLE. DLE-SYNC sequence. Both characters are written to the memory, but neither are included in the BCS calculation. DLE-ETX, DLE-ITB, DLE-ETB sequence, both characters are written to memory. The BCS is calculated only on the second character.
2–7	—	All zeros
8–15	DLE	DLE character

30.9 Sending and Receiving the Synchronization Sequence

The BISYNC channel can be programmed to send and receive a synchronization pattern defined in the DSR. GSMR_H[SYNL] defines pattern length, as shown in Table 30-7. The receiver synchronizes on this pattern. Unless SYNL is zero (external sync), the transmitter always sends the entire DSR contents, lsb first, before each frame—the chosen 4- or 8-bit pattern can be repeated in the lower-order bits.

Table 30-7. Receiver SYNC Pattern Lengths of the DSR

GSMR_H[SYNL] Setting	Bit Assignments															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	An external SYNC signal is used instead of the SYNC pattern in the DSR.															
01	4-Bit															
10	8-Bit															
11	16-Bit															

30.10 Handling Errors in the SCC BISYNC

The controller reports message transmit and receive errors using the channel BDs, error counters, and the SCCE. Modem lines can be directly monitored via the parallel port pins. Table 30-8 describes transmit errors.

Table 30-8. Transmit Errors

Error	Description
Transmitter Underrun	The channel stops sending the buffer, closes it, sets TxBD[UN], and generates aTXE interrupt if it is enabled. The channel resumes transmission after a RESTART TRANSMIT command is received. Underrun cannot occur between frames or during a DLE–*XXX* pair in transparent mode.
$\overline{\text{CTS}}$ Lost during Message Transmission	The channel stops sending the buffer, closes it, sets TxBD[CT], and generates a TXE interrupt if not masked. Transmission resumes when a RESTART TRANSMIT command is received.

Table 30-9 describes receive errors.

Table 30-9. Receive Errors

Error	Description
Overrun	The controller maintains a receiver FIFO for receiving data. The CP begins programming the SDMA channel (if the buffer is in external memory) and updating the CRC when the first byte is received in the Rx FIFO. If an Rx FIFO overrun occurs, the controller writes the received byte over the previously received byte. The previous character and its status bits are lost. The channel then closes the buffer, sets RxBD[OV], and generates the RXB interrupt if it is enabled. Finally, the receiver enters hunt mode.
$\overline{\text{CD}}$ Lost during Message Reception	The channel stops receiving, closes the buffer, sets RxBD[CD], and generates the RXB interrupt if not masked. This error has the highest priority. If the rest of the message is lost, no other errors are checked in the message. The receiver immediately enters hunt mode.
Parity	The channel writes the received character to the buffer and sets RxBD[PR]. The channel stops receiving, closes the buffer, sets RxBD[PR], and generates the RXB interrupt if it is enabled. The channel also increments PAREC and the receiver immediately enters hunt mode.
CRC	The channel updates the CR bit in the BD every time a character is received with a byte delay of eight serial clocks between the status update and the CRC calculation. When control character recognition is used to detect the end of the block and cause CRC checking, the channel closes the buffer, sets the CR bit in the BD, and generates the RXB interrupt if it is enabled.

30.11 BISYNC Mode Register (PSMR)

The PSMR is used as the BISYNC mode register, shown in Figure 30-5. PSMR[RBCS, RTR, RPM, TPM] can be modified on-the-fly.

	0–3	4–5	6	7	8	9	10–11	12–13	14–15
Field	NOS	CRC	RBCS	RTR	RVD	DRT	—	RPM	TPM
Reset	0								
R/W	R/W								
Addr	0x9_1A08 (PSMR1); 0x9_1A28 (PSMR2); 0x9_1A48 (PSMR3); 0x9_1A68 (PSMR4)								

Figure 30-5. Protocol-Specific Mode Register for BISYNC (PSMR)

Table 30-10 describes PSMR fields.

Table 30-10. PSMR Field Descriptions

Bits	Name	Description
0–3	NOS	Minimum number of SYN1–SYN2 pairs (defined in DSR) sent between or before messages.If NOS = 0000, one pair is sent. If NOS = 1111, 16 pairs are sent. The entire pair is always sent, regardless of how GSMR[SYNL] is set. NOS can be modified on-the-fly.
4–5	CRC	CRC selection. x0 Reserved. 01 CRC16 (BISYNC). X16 + X15 + X2 + 1. PRCRC and PTCRC should be initialized to all zeros or all ones before the channel is enabled. In either case, the transmitter sends the calculated CRC noninverted and the receiver checks the CRC against zero. Eight-bit data characters (without parity) are configured when CRC16 is chosen. 11 LRC (sum check). (BISYNC). For even LRC, initialize PRCRC and PTCRC to zeros before the channel is enabled; for odd LRC, they should be initialized to ones. **Note:** The receiver checks character parity when BCS is programmed to LRC and the receiver is not in transparent mode. The transmitter sends character parity when BCS is programmed to LRC and the transmitter is not in transparent mode. Use of parity in BISYNC assumes that 7-bit data characters are being used.
6	RBCS	Receive BCS. The receiver internally stores two BCS calculations separated by an eight serial clock delay to allow examination of a received byte to determine whether it should used in BCS calculation. 0 Disable receive BCS. 1 Enable receive BCS. Should be set (or reset) within the time taken to receive the following data byte. When RBCS is reset, BCS calculations exclude the latest fully received data byte. When RBCS is set, BCS calculations continue as normal.
7	RTR	Receiver transparent mode. 0 Normal receiver mode with SYNC stripping and control character recognition. 1 Transparent receiver mode. SYNCs, DLEs, and control characters are recognized only after a leading DLE character. The receiver calculates the CRC16 sequence even if it is programmed to LRC while in transparent mode. Initialize PRCRC to the CRC16 preset value before setting RTR.
8	RVD	Reverse data. 0 Normal operation. 1 Any portion of this SCC defined to operate in BISYNC mode operates by reversing the character bit order and sending the msb first.
9	DRT	Disable receiver while sending. DRT should not be set for typical BISYNC operation. 0 Normal operation. 1 As the SCC sends data, the receiver is disabled and gated by the internal $\overline{\text{RTS}}$ signal. This helps if the BISYNC channel is being configured onto a multidrop line and the user does not want to receive its own transmission. Although BISYNC usually uses a half-duplex protocol, the receiver is not actually disabled during transmission.
10–11	—	Reserved, should be cleared.

Table 30-10. PSMR Field Descriptions (continued)

Bits	Name	Description
12–13	RPM	Receiver parity mode. Selects the type of parity check that the receiver performs. RPM can be modified on-the-fly and is ignored unless CRC = 11 (LRC). Receive parity errors cannot be disabled but can be ignored. 00 Odd parity. The transmitter counts ones in the data word. If the sum is not odd, the parity bit is set to ensure an odd number. An even sum indicates a transmission error. 01 Low parity. If the parity bit is not low, a parity error is reported. 10 Even parity. An even number must result from the calculation performed at both ends of the line. 11 High parity. If the parity bit is not high, a parity error is reported.
14–15	TPM	Transmitter parity mode. Selects the type of parity the transmitter performs and can be modified on-the-fly. TPM is ignored unless CRC = 11 (LRC). 00 Odd parity. 01 Force low parity (always send a zero in the parity bit position). 10 Even parity. 11 Force high parity (always send a one in the parity bit position).

30.12 SCC BISYNC Receive BD (RxBD)

The CP uses BDs to report on each buffer received. It closes the buffer, generates a maskable interrupt, and starts receiving data into the next buffer after any of the following:

- A user-defined control character is received.
- An error is detected.
- A full receive buffer is detected.
- The ENTER HUNT MODE command is issued.
- The CLOSE RX BD command is issued.

Figure 30-6.Figure 30-6 shows the SCC BISYNC RxBD.

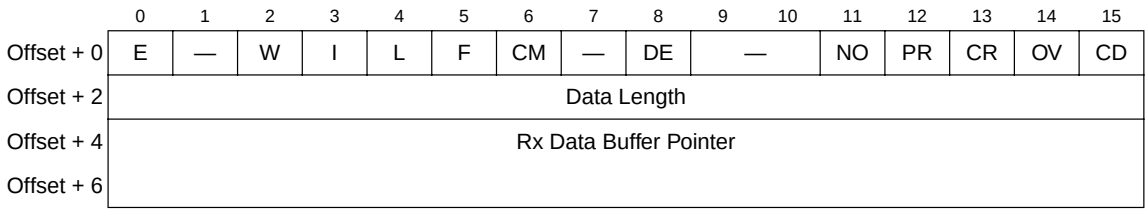


Figure 30-6. SCC BISYNC RxBD

Table 30-11Table 30-11. describes SCC BISYNC RxBD status and control fields.

Table 30-11. SCC BISYNC RxBD Status and Control Field Descriptions

Bits	Name	Description
0	E	Empty. 0 The buffer is full or stopped receiving because of an error. The core can read or write any fields of this RxBD. The CP does not use this BD as long as the E bit is zero. 1 The buffer is not full. The CP controls this BD and buffer. The core should not update this BD.
1	—	Reserved, should be cleared.
2	W	Wrap (last BD in table). 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to. The number of BDs in this table is determined by the W bit and by overall space constraints of the dual-port RAM.
3	I	Interrupt. 0 No interrupt is generated after this buffer is used. 1 SCCE[RXB] is set when the controller closes this buffer, which can cause an interrupt if it is enabled.
4	L	Last in frame. Set when this buffer is the last in a frame. If $\overline{CD}$ is negated in envelope mode or an error is received, one or more of the OV, CD, and DE bits are set. The controller writes the number of frame octets to the data length field. 0 Not the first buffer in the frame. 1 The first buffer in the frame.
5	F	First in frame. Set when this is the first buffer in a frame. 0 Not the first buffer in a frame. 1 First buffer in a frame
6	CM	Continuous mode. 0 Normal operation. 1 The CP does not clear E after this BD is closed; the buffer is overwritten when the CP accesses this BD next. However, E is cleared if an error occurs during reception, regardless of how CM is set.
7	—	Reserved, should be cleared.
8	DE	DPLL error. Set when a DPLL error occurs during reception. In decoding modes where a transition is should occur every bit, the DPLL error is set when a transition is missing.
9–10	—	Reserved, should be cleared.
11	NO	Rx non-octet-aligned frame. Set when a frame is received containing a number of bits not evenly divisible by eight.
12	PR	Parity error. Set when a character with parity error is received. Upon a parity error, the buffer is closed; thus, the corrupted character is the last byte of the buffer. A new Rx buffer receives subsequent data.
13	CR	Rx CRC error. Set when this frame contains a CRC error. Received CRC bytes are always written to the receive buffer.
14	OV	Overrun. Set when a receiver overrun occurs during frame reception.
15	CD	Carrier detect lost. Indicates when the carrier detect signal, $\overline{CD}$, is negated during frame reception.

Data length and buffer pointer fields are described in Section 27.3, “SCC Buffer Descriptors (BDs).” Data length represents the number of octets the CP writes into this buffer, including the BCS. For BISYNC mode, clear these bits. It is incremented each time a received character is written to the buffer.

30.13 SCC BISYNC Transmit BD (TxBD)

The CP arranges data to be sent on an SCC channel in buffers referenced by the channel TxBD table. The CP uses BDs to confirm transmission or indicate errors so the core knows buffers have been serviced. The user configures status and control bits before transmission, but the CP sets them after the buffer is sent.

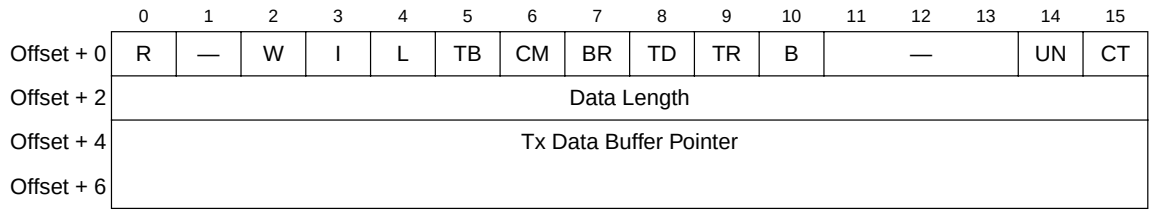


Figure 30-7. SCC BISYNC Transmit BD (TxBD)

Table 30-12 describes SCC BISYNC TxBD status and control fields.

Table 30-12. SCC BISYNC TxBD Status and Control Field Descriptions

Bits	Name	Description
0	R	Ready. 0 The buffer is not ready for transmission. The current BD and buffer can be updated. The CP clears R after the buffer is sent or after an error condition. 1 The user-prepared buffer has not been sent or is being sent. This BD cannot be updated while R = 1.
1	—	Reserved, should be cleared.
2	W	Wrap (last BD in table). 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CP sends data using the BD pointed to by TBASE. The number of TxBDs in this table is determined only by the W bit and overall space constraints of the dual-ported RAM.
3	I	Interrupt. 0 No interrupt is generated after this buffer is serviced. 1 SCCE[TXB] or SCCE[TXE] is set after the CP services this buffer, which can cause an interrupt.
4	L	Last in message. 0 The last character in the buffer is not the last character in the current block. 1 The last character in the buffer is the last character in the current block. The transmitter enters and stays in normal mode after sending the last character in the buffer and the BCS, if enabled.
5	TB	Transmit BCS. Valid only when the L bit is set. 0 Send an SYN1–SYN2 or idle sequence (specified in GSMR[RTSM]) after the last character in the buffer. 1 Send the BCS sequence after the last character. The controller also resets the BCS generator after sending the BCS.
6	CM	Continuous mode. 0 Normal operation. 1 The CP does not clear R after this BD is closed, so the buffer is resent when the CP next accesses this BD. However, R is cleared if an error occurs during transmission, regardless of how CM is set.

Table 30-12. SCC BISYNC TxBD Status and Control Field Descriptions (continued)

Bits	Name	Description
7	BR	BCS reset. Determines whether transmitter BCS accumulation is reset before sending the data buffer. 0 BCS accumulation is not reset. 1 BCS accumulation is reset before sending the data buffer.
8	TD	Transmit DLE. 0 No automatic DLE transmission can occur before the data buffer. 1 The transmitter sends a DLE character before sending the buffer, which saves writing the first DLE to a separate buffer in transparent mode. See TR for information on control characters.
9	TR	Transparent mode. 0 The transmitter enters and stays in normal mode after sending the buffer. The transmitter automatically inserts SYNCs if an underrun condition occurs. 1 The transmitter enters or stays in transparent mode after sending the buffer. It automatically inserts DLE–SYNC pairs if an underrun occurs (the controller finishes a buffer with L = 0 and the next BD is not available). It also checks all characters before sending them. If a DLE is detected, another DLE is sent automatically. Insert a DLE or program the controller to insert one before each control character. The transmitter calculates the CRC16 BCS even if PSMR[BCS] is programmed to LRC. Initialize PTCRC to CRC16 before setting TR.
10	B	BCS enable. 0 The buffer consists of characters that are excluded from BCS accumulation. 1 The buffer consists of characters that are included in BCS accumulation.
11–13	—	Reserved, should be cleared.
14	UN	Underrun. Set when the BISYNC controller encounters a transmitter underrun error while sending the associated data buffer. The CPM writes UN after it sends the associated buffer.
15	CT	$\overline{\text{CTS}}$ lost. The CP sets CT when $\overline{\text{CTS}}$ is lost during message transmission after it sends the data buffer.

Data length and buffer pointer fields are described in Section 27.3, “SCC Buffer Descriptors (BDs).” Although it is never modified by the CP, data length should be greater than zero. The CPM writes these fields after it finishes sending the buffer.

30.14 BISYNC Event Register (SCCE)/BISYNC Mask Register (SCCM)

The BISYNC controller uses the SCC event register (SCCE) to report events recognized by the BISYNC channel and to generate interrupts. When an event is recognized, the controller sets the corresponding SCCE bit. Interrupts are enabled by setting, and masked by clearing, the equivalent bits in the BISYNC mask register (SCCM). SCCE bits are reset by writing ones; writing zeros has no effect. Unmasked bits must be reset before the CP negates the internal interrupt request signal.

	0–2	3	4	5	6–7	8	9–10	11	12	13	14	15
Field	—	GLR	GLT	DCC	—	GRA	—	TXE	RCH	BSY	TXB	RXB
Reset	0000_0000_0000_0000											
R/W	R/W											
Addr	0x9_1A10 (SCCE1); 0x9_1A30 (SCCE2); 0x9_1A50 (SCCE3); 0x9_1A70 (SCCE4) 0x9_1A14 (SCCM1); 0x9_1A34 (SCCM2); 0x9_1A54 (SCCM3); 0x9_1A74 (SCCM4)											

Figure 30-8. BISYNC Event Register (SCCE)/BISYNC Mask Register (SCCM)

Table 30-13 describes SCCE and SCCM fields.

Table 30-13. SCCE/SCCM Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared.
3	GLR	Glitch on receive. Set when the SCC finds an Rx clock glitch.
4	GLT	Glitch on transmit. Set when the SCC finds a Tx clock glitch.
5	DCC	DPLL CS changed. Set when carrier sense status generated by the DPLL changes. Real-time status can be found in SCCS. This is not the $\overline{CD}$ status discussed elsewhere. Valid only when DPLL is used.
6–7	—	Reserved, should be cleared.
8	GRA	Graceful stop complete. Set as soon the transmitter finishes any message in progress when a GRACEFUL STOP TRANSMIT is issued (immediately if no message is in progress).
9–10	—	Reserved, should be cleared.
11	TXE	Tx Error. Set when an error occurs on the transmitter channel.
12	RCH	Receive character. Set when a character is received and written to the buffer.
13	BSY	Busy. Set when a character is received and discarded due to a lack of buffers. The receiver resumes reception after an ENTER HUNT MODE command.
14	TXB	Tx buffer. Set when a buffer is sent. TXB is set as the last bit of data or the BCS begins transmission.
15	RXB	Rx buffer. Set when the CPM closes the receive buffer on the BISYNC channel.

30.15 SCC Status Registers (SCCS)

The SCC status (SCCS) register, as seen in Figure 30-9, allows real-time monitoring of RXD. The real-time status of $\overline{CTS}$ and $\overline{CD}$ are part of the parallel I/O.

	0–5	6	7
Field	—	CS	—
Reset	0000_0000		
R/W	R		
Addr	0x9_1A17 (SCCS1); 0x9_1A37 (SCCS2); 0x9_1A57 (SCCS3); 0x9_1A77 (SCCS4)		

Figure 30-9. SCC Status Registers (SCCS)

Table 30-14 describes SCCS fields.

Table 30-14. SCCS Field Descriptions

Bits	Name	Description
0–5	—	Reserved, should be cleared.
6	CS	Carrier sense (DPLL). Shows the real-time carrier sense of the line as determined by the DPLL. 0 The DPLL does not sense a carrier. 1 The DPLL senses a carrier.
7	—	Reserved, should be cleared.

30.16 Programming the SCC BISYNC Controller

Software has two ways to handle data received by the BISYNC controller. The simplest is to allocate single-byte receive buffers, request an interrupt on reception of each buffer, and implement BISYNC protocol entirely in software on a byte-by-byte basis. This flexible approach can be adapted to any BISYNC implementation. The obvious penalty is the overhead caused by interrupts on each received character.

A more efficient method is to prepare and link multi-byte buffers in the RxBD table and use software to analyze the first two to three bytes of the buffer to determine the type of block received. When this is determined, reception continues without further software intervention until it encounters a control character, which signifies the end of the block and causes software to revert to byte-by-byte reception.

To accomplish this, set SCCM[RCH] to enable an interrupt on every received byte so software can analyze each byte. After analyzing the initial characters of a block, either set PSMR[RTR] or issue a RESET BCS CALCULATION command. For example, if a DLE-STX is received, enter transparent mode. By setting the appropriate PSMR bit, the controller strips the leading DLE from DLE-character sequences. Thus, control characters are recognized only when they follow a DLE character. PSMR[RTR] should be cleared after a DLE-ETX is received.

Alternatively, after an SOH is received, a RESET BCS CALCULATION should be issued to exclude SOH from BCS accumulation and reset the BCS. Notice that PSMR[RBCS] is not needed because the controller automatically excludes SYNCs and leading DLEs.

After the type of block is recognized, SCCE[RCH] should be masked. The core does not interrupt data reception until the end of the current block, which is indicated by the reception of a control character matching the one in the receive control character table. Using Table 30-15, the control character table should be set to recognize the end of the block.

Table 30-15. Control Characters

Control Characters	E	B	H
ETX	0	1	1
ITB	0	1	0
ETB	0	1	1
ENQ	0	0	0
Next entry	0	X	X

After ETX, a BCS is expected; then the buffer should be closed. HUNT MODE should be entered when a line turnaround occurs. ENQ characters are used to stop sending a block and to designate the end of the block for a receiver, but no CRC is expected. After control character reception, set SCCM[RCH] to reenable interrupts for each byte of data received.

30.17 SCC BISYNC Programming Example

This BISYNC controller initialization example for SCC2 uses an external clock. The controller is configured with RTS2, CTS2, and CD2 active. Both the receiver and transmitter use CLK3.

1. Configure port D pins to enable TXD2 and RXD2. Set PPARD[27,28] and PDIRD[27] and clear PDIRD[28] and PSORD[27,28].
2. Configure ports C and D pins to enable RTS2, CTS2 and CD2. Set PPARD[26], PPARC[12,13] and PDIRD[26] and clear PDIRC[12,13], PSORC[12,13], and PSORD[26].
3. Configure port C pin 29 to enable the CLK3 pin. Set PPARC[29] and clear PDIRC[29] and PSORC[29].
4. Connect CLK3 to SCC2 using the CPM mux. Write 0b110 to CMXSCR[R2CS] and CMXSCR[T2CS].
5. Connect the SCC2 to the NMSI (its own set of pins). Clear CMXSCR[SC2].
6. Assuming one RxBD at the beginning of dual-port RAM followed by one TxBD, write RBASE with 0x0000 and TBASE with 0x0008.
7. Write 0x04a1_0000 to CPCR to execute INIT RX AND TX PARAMETERS. This updates RBPTR and TBPTR to the new values of RBASE and TBASE.
8. Write RFCR and TFCR with 0x10 for normal operation.
9. Write MRBLR with the maximum number of bytes per receive buffer. For this case, assume 16 bytes, so MRBLR = 0x0010.
10. Write PRCRC with 0x0000 to comply with CRC16.
11. Write PTCRC with 0x0000 to comply with CRC16.

12. Clear PAREC for clarity.
13. Write BSYNC with 0x8033, assuming a SYNC value of 0x33.
14. Write DSR with 0x3333.
15. Write BDLE with 0x8055, assuming a DLE value of 0x55.
16. Write CHARACTER1 with 0x6077, assuming ETX = 0x77.
17. Write CHARACTER2–8 with 0x8000. They are not used.
18. Write RCCM with 0xE0FF. It is not used.
19. Initialize the RxBD and assume the data buffer is at 0x00001000 in main memory. Then write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x00001000 to RxBD[Buffer Pointer].
20. Initialize the TxBD and assume the Tx data buffer is at 0x00002000 in main memory and contains five 8-bit characters. Then write 0xBD20 to TxBD[Status and Control] 0x0005 to TxBD[Data Length], and 0x00002000 to TxBD[Buffer Pointer]. Note that ETX character should be written at the end of user's data.
21. Write 0xFFFF to SCCE to clear any previous events.
22. Write 0x0013 to SCCM to enable the TXE, TXB, and RXB interrupts.
23. Write 0x0040_0000 to the SIU interrupt mask register low (SIMR_L) so the SMC1 can generate a system interrupt. Initialize SIU interrupt pending register low (SIPNR_L) by writing 0xFFFF_FFFF to it.
24. Write 0x0000002C to GSMR_H2 to configure a small receive FIFO width.
25. Write 0x00000008 to GSMR_L2 to configure $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ to automatically control transmission and reception (DIAG bits) and the BISYNC mode. Notice that the transmitter (ENT) and receiver (ENR) are not yet enabled.
26. Set PSMR to 0x0600 to configure CRC16, CRC checking on receive, and normal operation (not transparent).
27. Write 0x00000038 to GSMR_L2 to enable the SCC2 transmitter and receiver. This additional write ensures that ENT and ENR are enabled last.

Write 0x00000038 to GSMR_L2 to enable the SCC2 transmitter and receiver. This additional write ensures that ENT and ENR are enabled last. After five bytes are sent, the TxBD is closed. The buffer is closed after 16 bytes are received. Any received data beyond 16 bytes causes a busy (out-of-buffers) condition since only one RxBD is prepared.

Chapter 31
SCC Transparent Mode

Transparent mode (also called totally transparent or promiscuous mode) provides a clear channel on which the SCC can send or receive serial data without bit-level manipulation. Software implements protocols run over transparent mode. An SCC in transparent mode functions as a high-speed serial-to-parallel and parallel-to-serial converter.

Transparent mode can be used for serially moving data that requires no superimposed protocol, for applications that require serial-to-parallel and parallel-to-serial conversion for communication among chips on the same board, and for applications that require data to be switched without interfering with the protocol encoding itself, such as when data from a high-speed time-multiplexed serial stream is multiplexed into low-speed data streams. The concept is to switch the data path without altering the protocol encoded on that data path.

Transparent mode is configured in the GSMR; see Section 27.2, “General SCC Mode Registers (GSMR1–GSMR4).” Transparent mode is selected in GSMR_H[TTX, TRX] for the transmitter and receiver, respectively. Setting both bits enables full-duplex transparent operation. If only one is set, the other half of the SCC uses the protocol specified in GSMR_L[MODE]. This allows loop-back modes to DMA data from one memory location to another while data is converted to a specific serial format.

The SCC operations are asynchronous with the core and can be synchronous or asynchronous with other SCCs. Each clock can be supplied from the internal baud rate generator bank, DPLL output, or external pins.

The SCC can work with the time-slot assigner (TSA) or nonmultiplexed serial interface (NMSI) and supports modem lines with the general-purpose I/O pins. Data can be transferred either the msb or lsb first in each octet.

31.1 Features

The following list summarizes the main features of the SCC in transparent mode:

- Flexible buffers
- Automatic SYNC detection on receive
- CRCs can be sent and received
- Reverse data mode

- Another protocol can be performed on the other half of the SCC
- MC68360-compatible SYNC options

31.2 SCC Transparent Channel Frame Transmission Process

The transparent transmitter is designed to work almost no intervention from the core. When the core enables the SCC transmitter in transparent mode, it starts sending idles, which are logic high or encoded ones, as programmed in GSMR_L[TEND]. The SCC polls the first BD in the TxBD table. When there is a message to send, the SCC fetches data from memory, loads the transmit FIFO, and waits for transmitter synchronization, which is achieved with $\overline{\text{CTS}}$ or by waiting for the receiver to achieve synchronization, depending on GSMR_H[TXSY]. Transmission begins when transmitter synchronization is achieved.

When all BD data has been sent, if TxBD[L] is set, the SCC writes the message status bits into the BD, clears TxBD[R], and sends idles until the next BD is ready. If it is ready, some idles are still sent. The transmitter resumes sending only after it achieves synchronization.

If TxBD[L] is cleared when the end of the BD is reached, only TxBD[R] is cleared and the transmitter moves immediately to the next buffer to begin transmission with no gap on the serial line between buffers. Failure to provide the next buffer in time causes a transmit underrun which sets SCCE[TXE].

In both cases, an interrupt is issued according to TxBD[I]. By appropriately setting TxBD[I] in each BD, interrupts are generated after each buffer or group of buffers is sent. The SCC then proceeds to the next BD in the table and any whole number of bytes can be sent. If GSMR_H[REVD] is set, the bit order of each byte is reversed before being sent; the msb of each octet is sent first.

Setting GSMR_H[TFL] makes the transmit FIFO smaller and reduces transmitter latency, but it can cause transmitter underruns at higher transmission speeds. An optional CRC, selected in GSMR_H[TCRC], can be appended to each transparent frame if it is enabled in the TxBD.

When the time-slot assigner (TSA) is used with a transparent-mode channel, synchronization is provided by the TSA. There is a start-up delay for the transmitter, but delays will always be some whole number of complete TSA frames. This means that *n*-byte transmit buffers can be mapped directly into *n*-byte time slots in the TSA frames.

31.3 SCC Transparent Channel Frame Reception Process

When the core enables the SCC receiver in transparent mode, it waits to achieve synchronization before data is received. The receiver can be synchronized to the data by a synchronization pulse or SYNC pattern.

After a buffer is full, the SCC clears RxBD[E] and generates a maskable interrupt if RxBD[I] is set. It moves to the next RxBD in the table and begins moving data to its buffer. If the next buffer is not available, SCCE[BSY] signifies a busy signal that can generate a maskable interrupt. The receiver reverts to hunt mode when an ENTER HUNT MODE command or an error is received. If GSMR_H[REVD] is set, the bit order of each byte is reversed before it is written to memory.

Setting GSMR_H[RFW] reduces receiver latency by making the receive FIFO smaller, which may cause receiver overruns at higher transmission speeds. The receiver always checks the CRC of the received frame, according to GSMR_H[TCRC]. If a CRC is not required, resulting errors can be ignored.

31.4 Achieving Synchronization in Transparent Mode

Once the SCC transmitter is enabled for transparent operation, the TxBD is prepared and the transmit FIFO is preloaded by the SDMA channel, another process must occur before data can be sent. It is called transmit synchronization. Similarly, once the SCC receiver is enabled for transparent operation in the GSMR and the RxBD is made empty for the SCC, receive synchronization must occur before data can be received. An in-line synchronization pattern or an external synchronization signal can provide bit-level control of the synchronization process when sending or receiving.

31.4.1 Synchronization in NMSI Mode

This section describes synchronization in NMSI mode.

31.4.1.1 In-Line Synchronization Pattern

The transparent channel can be programmed to receive a synchronization pattern. This pattern is defined in the data synchronization register, DSR; see Section 27.2.2, “Data Synchronization Register (DSR).” Pattern length is specified in GSMR_H[SYNL], as shown in Figure 31-1. See also Section 27.2, “General SCC Mode Registers (GSMR1–GSMR4).”

Table 31-1. Receiver SYNC Pattern Lengths of the DSR

GSMR_H[SYNL] Setting	Bit Assignments															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	An external SYNC signal is used instead of the SYNC pattern in the DSR.															
01	4-Bit															
10	8-Bit															
11	16-Bit															

If a 4-bit SYNC is selected, reception begins as soon as these four bits are received, beginning with the first bit following the 4-bit SYNC. The transmitter synchronizes on the receiver pattern if GSMR_H[RSYN] = 1.

Note that the transparent controller does not automatically send the synchronization pattern; therefore, the synchronization pattern must be included in the transmit buffer.

31.4.1.2 External Synchronization Signals

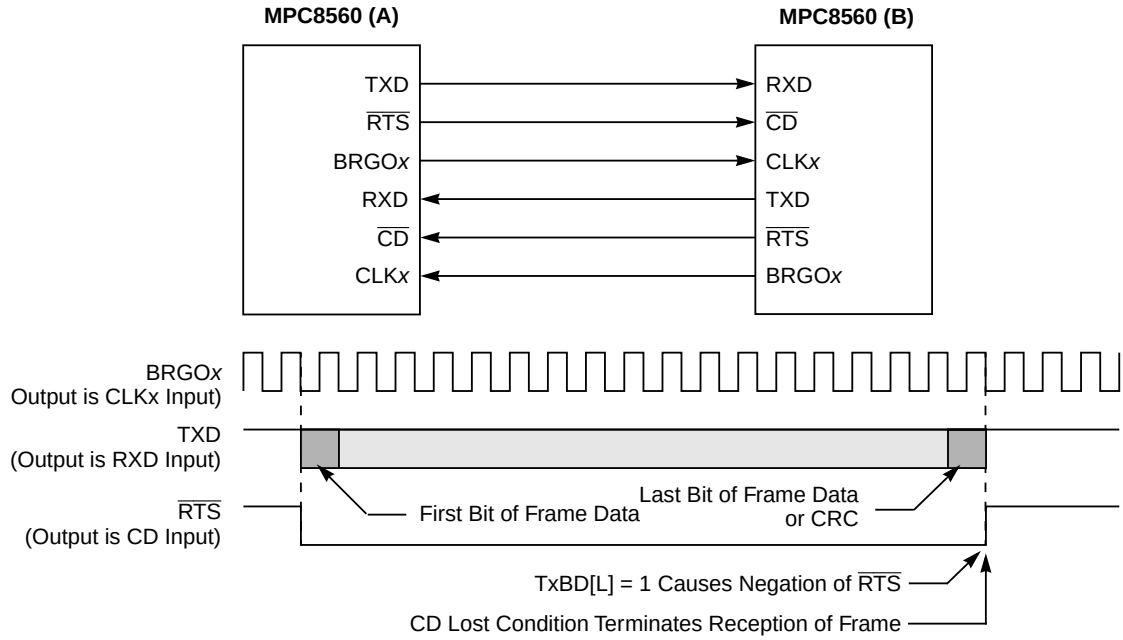
If GSMR_H[SYNL] is 0b00, the transmitter uses $\overline{\text{CTS}}$ and the receiver uses $\overline{\text{CD}}$ to begin the sequence. These signals share two options—pulsing and sampling.

GSMR_H[CDP] and GSMR_H[CTSP] determine whether $\overline{\text{CD}}$ or $\overline{\text{CTS}}$ need to be asserted only once to begin reception/transmission or whether they must remain asserted for the duration of the transparent frame. Pulse operation allows an uninterrupted stream of data. However, use envelope mode to identify frames of transparent data.

The sampling option determines the delay between $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ being asserted and the resulting action by the SCC. Assume either that these signals are asynchronous to the data and internally synchronized by the SCC or that they are synchronous to the data with faster operation. This option allows $\overline{\text{RTS}}$ of one SCC to be connected to $\overline{\text{CD}}$ of another SCC and to have the data synchronized and bit aligned. It is also an option to link the transmitter synchronization to the receiver synchronization. Diagrams for the pulse/envelope and sampling options are shown in Section 31.4, “Achieving Synchronization in Transparent Mode.”

31.4.1.2.1 External Synchronization Example

Figure 31-1 shows synchronization using external signals.



Notes:
- 1. Each MPC8560 generates its own transmit clocks. If the transmit and receive clocks are the same, one MPC8560 can generate transmit and receive clocks for the other MPC8560. For example, CLK*x* on MPC8560 (B) could be used to clock the transmitter and receiver.
- 2. $\overline{\text{CTS}}$ should be configured as always asserted in the parallel I/O or connected to ground externally.
- 3. The required GSMR configurations are DIAG= 00, CTSS=1, CTSP is a “don’t care”, CDS=1, CDP=0, TTX=1, and TRX=1. REVD and TCRC are application-dependent.
- 4. The transparent frame contains a CRC if TxBD[TC] is set.

Figure 31-1. Sending Transparent Frames between MPC8560s

MPC8560(A) and MPC8560(B) exchange transparent frames and synchronize each other using $\overline{\text{RTS}}$ and $\overline{\text{CD}}$. However, $\overline{\text{CTS}}$ is not required because transmission begins at any time. Thus, $\overline{\text{RTS}}$ is connected directly to the other MPC8560 $\overline{\text{CD}}$ pin. GSMR_H[RSYN] is not used and transmission and reception from each MPC8560 are independent.

31.4.1.3 Transparent Mode without Explicit Synchronization

If there is no need to synchronize the transparent controller at a specific point, the user can ‘fake’ synchronization in one of the following ways:

- Tie a parallel I/O pin to the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ lines. Then, after enabling the receiver and transmitter, provide a falling edge by manipulating the I/O pin in software.
- Enable the receiver and transmitter for the SCC in loopback mode and then change GSMR_L[DIAG] to 0b00 while the transmitter and receiver and enabled.

31.4.2 Synchronization and the TSA

A transparent-mode SCC using the time-slot assigner can synchronize either on a user-defined inline pattern or by inherent synchronization.

Note that when using the TSA, a newly-enabled transmitter sends from 10 to 15 frames of idles before sending the actual transparent data due to startup requirements of the TDM. Therefore, when loopback testing through the TDM, expect to receive several bytes of 0xFF before the actual data.

31.4.2.1 Inline Synchronization Pattern

The receiver can be programmed to begin receiving data into the receive buffers only after a specified data pattern arrives. To synchronize on an inline pattern:

- Set GSMR_H[SYNL].
- Program the DSR with the desired pattern.
- Clear GSMR_H[CDP].
- Set GSMR_H[CTSP, CTSS, CDS].

If GSMR_H[TXSY] is also used, the transmitter begins transmission eight clocks after the receiver achieves synchronization.

31.4.2.2 Inherent Synchronization

Inherent synchronization assumes synchronization by default when the channel is enabled; all data sent from the TDM to the SCC is received. To implement inherent synchronization:

- Set GSMR_H[CDP, CDS, CTSP, CTSS].

If these bits are not set, the received bit stream will be bit-shifted. The SCC loses the first received bit because $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ are treated as asynchronous signals.

31.4.3 End of Frame Detection

An end of frame cannot be detected in the transparent data stream since there is no defined closing flag in transparent mode. Therefore, if framing is needed, the user must use the $\overline{\text{CD}}$ line to alert the transparent controller of an end of frame.

31.5 CRC Calculation in Transparent Mode

The CRC calculations follow the ITU/IEEE standard. The CRC is calculated on the transmitted data stream; that is, from lsb to msb for non-bit-reversed (GSMR_H[REVD] = 0) and from msb to lsb for bit-reversed (GSMR_H[REVD] = 1) transmission. The appended CRC is sent msb to lsb.

When receiving, the CRC is calculated as the incoming bits arrive. The optional reversal of data (GSMR_H[REVD] = 1) is done just before data is stored in memory (after the CRC calculation).

31.6 SCC Transparent Parameter RAM

For transparent mode, the protocol-specific area of the SCC parameter RAM is mapped as in Table 31-2Table 31-2..

Table 31-2. SCC Transparent Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x 30	CRC_P	Long	CRC preset for totally transparent. For the 16-bit CRC-CCITT, initialize with 0x0000_FFFF. For the 32-bit CRC-CCITT, initialize with 0xFFFF_FFFF and for the CRC-16, initialize with ones (0x0000_FFFF) or zeros (0x0000_0000).
0x 34	CRC_C	Long	CRC constant for totally transparent receiver. For the 16-bit CRC-CCITT, initialize with 0x0000_F0B8. For the 32-bit CRC-CCITT, CRC_C initialize with 0xDEBB_20E3 and for the CRC-16, which is normally used with BISYNC, initialize with 0x0000_0000.

[1] From SCC base address. See Section 27.4.1, “SCC Base Addresses.”

CRC_P and CRC_C overlap with the CRC parameters for the HDLC-based protocols. However, this overlap is not detrimental since the CRC constant is used only for the receiver and the CRC preset is used only for the transmitter, so only one entry is required for each. Thus, the user can choose an HDLC transmitter with a transparent receiver or a transparent transmitter with an HDLC receiver.

31.7 SCC Transparent Commands

The following transmit and receive commands are issued to the CP command register. Table 31-3. Table 31-3 describes transmit commands.

Table 31-3. Transmit Commands

Command	Description
STOP TRANSMIT	After hardware or software is reset and the channel is enabled in the GSMR, the channel is in transmit enable mode and starts polling the first BD every 64 clocks (or immediately if TODR[TOD] = 1). STOP TRANSMIT disables frame transmission on the transmit channel. If the transparent controller receives the command during frame transmission, transmission is aborted after a maximum of 64 additional bits and the transmit FIFO is flushed. The current TxBD pointer (TBPTR) is not advanced, no new BD is accessed and no new buffers are sent for this channel. The transmitter will send idles.
GRACEFUL STOP TRANSMIT	Stops transmission smoothly, rather than abruptly, in much the same way that the regular STOP TRANSMIT command stops. It stops transmission after the current frame finishes or immediately if no frame is being sent. A transparent frame is not complete until a BD with TxBD[L] set has its buffer completely sent. SCCE[GRA] is set once transmission stops; transmit parameters and their BDs can then be modified. The current TxBD pointer (TBPTR) advances to the next TxBD in the table. Transmission resumes once TxBD[R] is set and a RESTART TRANSMIT command is issued.

Table 31-3. Transmit Commands (continued)

Command	Description
RESTART TRANSMIT	Reenables transmission of characters on the transmit channel. The transparent controller expects it after a STOP TRANSMIT command is issued (at which point the channel is disabled in SCCM), after a GRACEFUL STOP TRANSMIT command is issued, or after a transmitter error. The transparent controller resumes transmission from the current TBPTR in the channel TxBD table.
INIT TX PARAMETERS	Initializes all transmit parameters in the serial channel parameter RAM to reset state. Issue only when the transmitter is disabled. INIT TX AND RX PARAMETERS resets receive and transmit parameters.

Table 31-4 describes receive commands.

Table 31-4. Receive Commands

Command	Description
ENTER HUNT MODE	After hardware or software is reset and the channel is enabled, the channel is in receive enable mode and uses the first BD in the table. ENTER HUNT MODE forces the transparent receiver to the current frame and enter hunt mode where the transparent controller waits for the synchronization sequence. After receiving the command, the current buffer is closed. Further data reception uses the next BD.
CLOSE RXBD	Forces the SCC to close the RxBD if it is being used and to use the next BD for any subsequently received data. If the SCC is not receiving data, no action is taken by this command.
INIT RX PARAMETERS	Initializes all receive parameters in this serial channel parameter RAM to reset state. Issue only when the receiver is disabled. INIT TX AND RX PARAMETERS resets receive and transmit parameters.

31.8 Handling Errors in the Transparent Controller

The SCC reports message reception and transmission errors using the channel buffer descriptors, the error counters, and SCCE. Table 31-5 describes transmit errors.

Table 31-5. Transmit Errors

Error	Description
Transmitter Underrun	When this occurs, the channel stops sending the buffer, closes it, sets TxBD[UN], and generates a TXE interrupt if it is enabled. Transmission resumes after a RESTART TRANSMIT command is received. Underrun occurs after a transmit frame for which TxBD[L] was not set. In this case, only SCCE[TXE] is set. Underrun cannot occur between transparent frames.
$\overline{\text{CTS}}$ Lost During Message Transmission	When this occurs, the channel stops sending the buffer, closes it, sets TxBD[CT], and generates the TXE interrupt if it is enabled. The channel resumes sending after RESTART TRANSMIT is received.

Table 31-6 describes receive errors.

Table 31-6. Receive Errors

Error	Description
Overrun	The SCC maintains a receive FIFO. The CPM starts programming the SDMA channel if the buffer is in external memory and updating the CRC when 8 or 32 bits are received in the FIFO as determined by GSMR_H[RFW]. If a FIFO overrun occurs, the SCC writes the received byte over the previously received byte. The previous character and its status bits are lost. Afterwards, the channel closes the buffer, sets OV in the BD, and generates the RXB interrupt if it is enabled. The receiver immediately enters hunt mode.
$\overline{\text{CD}}$ Lost During Message Reception	When this occurs, the channel stops receiving messages, closes the buffer, sets RxBD[CD], and generates the RXB interrupt if it is enabled. This error has highest priority. The rest of the message is lost, and no other errors are checked in the message. The receiver immediately enters hunt mode.

31.9 Transparent Mode and the PSMR

The protocol-specific mode register (PSMR) is not used by the transparent controller because all transparent mode selections are made in the GSMR. If only half of an SCC (transmitter or receiver) is running the transparent protocol, the other half (receiver or transmitter) can support another protocol. In such a case, use the PSMR for the non-transparent protocol.

31.10 SCC Transparent Receive Buffer Descriptor (RxBD)

The CPM reports information about the received data for each buffer using an RxBD (see Figure 31-2), closes the current buffer, generates a maskable interrupt, and starts receiving data into the next buffer after one of the following occurs:

- An error is detected.
- A full receive buffer is detected.
- An ENTER HUNT MODE command is issued.
- A CLOSE RXBD command is issued.

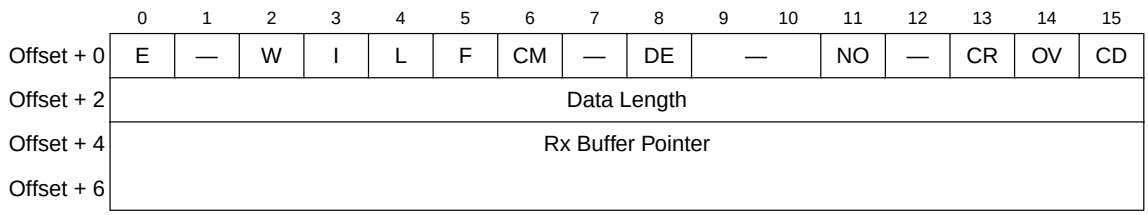


Figure 31-2. SCC Transparent Receive Buffer Descriptor (RxBD)

Table 31-7 describes RxBD status and control fields.

Table 31-7. SCC Transparent RxBD Status and Control Field Descriptions

Bits	Name	Description
0	E	Empty. 0 The buffer is full or stopped receiving data because an error occurred. The core can read or write to any fields of this RxBD. The CPM does not use this BD when RxBD[E] is zero. 1 The buffer is not full. This RxBD and buffer are owned by the CPM. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared.
2	W	Wrap (final BD in table). 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CPM receives data into the first BD that RBASE points to. The number of BDs in this table is determined only by RxBD[W] and overall space constraints of the dual-port RAM.
3	I	Interrupt. 0 No interrupt is generated after this buffer is used. 1 When this buffer is closed by the transparent controller, the SCCE[RXB] is set. SCCE[RXB] can cause an interrupt if it is enabled.
4	L	Last in frame. Set by the transparent controller when this buffer is the last in a frame, which occurs when $\overline{CD}$ is negated (if GSMR_H[CDP] = 0) or an error is received. If an error is received, one or more of RxBD[OV, CD, DE] are set. Note that the SCC transparent controller writes the number of buffer (not frame) octets to the last BD's data length field. 0 Not the last buffer in a frame. 1 Last buffer in a frame.
5	F	First in frame. The transparent controller sets F when this buffer is the first in the frame: 0 Not the first buffer in a frame. 1 First buffer in a frame.
6	CM	Continuous mode. 0 Normal operation. 1 The CPM does not clear RxBD[E] after this BD is closed, letting the buffer be overwritten when the CPM next accesses this BD. However, RxBD[E] is cleared if an error occurs during reception, regardless of how CM is set.
7	—	Reserved, should be cleared.
8	DE	DPLL error. Set by the transparent controller when a DPLL error occurs as this buffer is received. In decoding modes, where a transition is promised every bit, DE is set when a missing transition occurs. If a DPLL error occurs, no other error checking is performed.
9–10	—	Reserved, should be cleared.
11	NO	Rx non-octet. Set when a frame containing a number of bits not exactly divisible by eight is received.
12	—	Reserved, should be cleared.
13	CR	CRC error indication bits. Indicates that this frame contains a CRC error. The received CRC bytes are always written to the receive buffer. CRC checking cannot be disabled, but it can be ignored.
14	OV	Overrun. Indicates that a receiver overrun occurred during buffer reception.
15	CD	Carrier detect lost. Indicates when $\overline{CD}$ is negated during buffer reception.

Data length and buffer pointer fields are described in Section 27.3, "SCC Buffer Descriptors (BDs)." The Rx buffer pointer must be divisible by four, unless

GSMR_H[RFW] is set to 8 bits wide, in which case the pointer can be even or odd. The buffer can reside in internal or external memory.

31.11 SCC Transparent Transmit Buffer Descriptor (TxBD)

Data is sent to the CPM for transmission on an SCC channel by arranging it in buffers referenced by the TxBD table. The CPM uses BDs to confirm transmission or indicate error conditions so the processor knows buffers have been serviced. Status and control bits must be prepared before transmission; they are set by the CPM after the buffer is sent. The TxBD is shown in Figure 31-3.

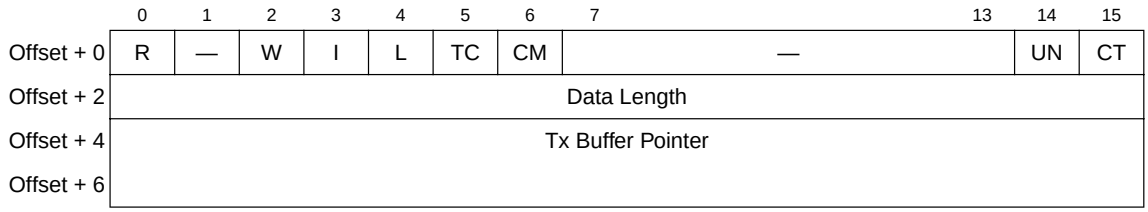


Figure 31-3. SCC Transparent Transmit Buffer Descriptor (TxBD)

Figure 31-8 describes SCC Transparent TxBD status and control fields.

Table 31-8. SCC Transparent TxBD Status and Control Field Descriptions

Bits	Name	Description
0	R	Ready. 0 The buffer is not ready for transmission. The BD and buffer can be updated. The CPM clears R after the buffer is sent or after an error is encountered. 1 The user-prepared buffer is not sent yet or is being sent. This BD cannot be updated while R = 1.
1	—	Reserved, should be cleared.
2	W	Wrap (final BD in table). 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CPM receives incoming data into the first BD that TBASE points to. The number of TxBDs in this table is determined only by TxBD[W] and overall space constraints of the dual-port RAM.
3	I	Interrupt. Note that clearing this bit does not disable all SCCE[TXE] events. 0 No interrupt is generated after this buffer is serviced. 1 When the CPM services this buffer, SCCE[TXB] or SCCE[TXE] is set. These bits can cause interrupts if they are enabled.
4	L	Last in message. 0 The last byte in the buffer is not the last byte in the transmitted transparent frame. Data from the next transmit buffer is sent immediately after the last byte of this buffer. 1 The last byte in the buffer is the last byte in the transmitted transparent frame. After this buffer is sent, the transmitter requires synchronization before the next buffer is sent.
5	TC	Transmit CRC. 0 No CRC sequence is sent after this buffer. 1 A frame check sequence defined by GSMR_H[TCRC] is sent after the last byte of this buffer.

Table 31-8. SCC Transparent TxBD Status and Control Field Descriptions

Bits	Name	Description
6	CM	Continuous mode. 0 Normal operation. 1 The CPM does not clear TxBD[R] after this BD is closed, so the buffer is automatically resent when the CPM accesses this BD next. However, TxBD[R] is cleared if an error occurs during transmission, regardless of how CM is set.
7–13	—	Reserved, should be cleared.
14	UN	Underrun. Set when the SCC encounters a transmitter underrun condition while sending the buffer.
15	CT	CTS lost. Indicates the $\overline{\text{CTS}}$ was lost during frame transmission.

Data length and buffer pointer fields are described in Section 27.3, “SCC Buffer Descriptors (BDs).” Although it is never modified by the CP, data length should be greater than zero. The buffer pointer can be even or odd and can reside in internal or external memory.

31.12 SCC Transparent Event Register (SCCE)/Mask Register (SCCM)

When the SCC is in transparent mode, the SCC event register (SCCE) functions as the transparent event register to report events recognized by the transparent channel and to generate interrupts. When an event is recognized, the transparent controller sets the corresponding SCCE bit. Interrupts are enabled by setting, and masked by clearing, the equivalent bits in the transparent mask register (SCCM).

Event bits are reset by writing ones; writing zeros has no effect. All unmasked bits must be reset before the CP clears the internal interrupt request to the SIU interrupt controller. Figure 31-4 shows the event and mask registers.

	0–2	3	4	5	6–7	8	9–10	11	12	13	14	15
Field	—	GLR	GLT	DCC	—	GRA	—	TXE	—	BSY	TXB	RXB
Reset	0000_0000_0000_0000											
R/W	R/W											
Addr	0x9_1A10 (SCCE1); 0x9_1A30 (SCCE2); 0x9_1A50 (SCCE3); 0x9_1A70 (SCCE4) 0x9_1A14 (SCCM1); 0x9_1A34 (SCCM2); 0x9_1A54 (SCCM3); 0x9_1A74 (SCCM4)											

Figure 31-4. SCC Transparent Event Register (SCCE)/Mask Register (SCCM)

Table 31-9 describes SCCE/SCCM fields.

Table 31-9. SCCE/SCCM Field Descriptions

Bits	Name	Description
0–2	—	Reserved, should be cleared.
3	GLR	Glitch on Rx. Set when the SCC finds a glitch on the receive clock.
4	GLT	Glitch on Tx. Set when the SCC finds a glitch on the transmit clock.
5	DCC	DPLL CS changed. Set when the DPLL-generated carrier sense status changes (valid only when the DPLL is used). Real-time status can be read in SCCS. This is not the $\overline{CD}$ status mentioned elsewhere.
6–7	—	Reserved, should be cleared.
8	GRA	Graceful stop complete. Set when a graceful stop initiated by completes as soon as the transmitter finishes any frame in progress when the GRACEFUL STOP TRANSMIT command was issued. Immediately if no frame was in progress when the command was issued.
9–10	—	Reserved, should be cleared.
11	TXE	Tx error. Set when an error occurs on the transmitter channel.
12	—	Reserved, should be cleared.
13	BSY	Busy condition. Set when a byte or word is received and discarded due to a lack of buffers. The receiver resumes reception after it gets an ENTER HUNT MODE command.
14	TXB	Tx buffer. Set no sooner than when the last bit of the last byte of the buffer begins transmission, assuming L is set in the TxBD. If it is not, TXB is set when the last byte is written to the transmit FIFO.
15	RXB	Rx buffer. Set when a complete buffer was received on the SCC channel, no sooner than two serial clocks after the last bit of the last byte in which the buffer is received on RXD.

31.13 SCC Status Register in Transparent Mode (SCCS)

The SCC status register (SCCS), shown in Figure 31-5, allows monitoring of real-time status conditions on the RXD line. The real-time status of $\overline{CTS}$ and $\overline{CD}$ are part of the parallel I/O.

	0 — 5	6	7
Field	—	CS	—
Reset	0000_0000		
R/W	R		
Addr	0x9_1A17 (SCCS1); 0x9_1A37 (SCCS2); 0x9_1A57 (SCCS3); 0x9_1A77 (SCCS4)		

Figure 31-5. SCC Status Register in Transparent Mode (SCCS)

Table 31-10 describes SCCS fields.

Table 31-10. SCCS Field Descriptions

Bits	Name	Description
0–5	—	Reserved, should be cleared.
6	CS	Carrier sense (DPLL). Shows the real-time carrier sense of the line as determined by the DPLL. 0 The DPLL does not sense a carrier. 1 The DPLL senses a carrier.
7	—	Reserved, should be cleared.

31.14 SCC2 Transparent Programming Example

The following initialization sequence enables the transmitter and receiver, which operate independently of each other. They implement the connection shown on MPC8560(B) in Figure 31-1.

The transmit and receive clocks are externally provided to MPC8560(B) using CLK3. SCC2 is used. The transparent controller is configured with the $\overline{\text{RTS2}}$ and $\overline{\text{CD2}}$ pins active and $\overline{\text{CTS2}}$ is configured to be grounded internally. A 16-bit CRC-CCITT is sent with each transparent frame. The FIFOs are configured for fast operation.

1. Configure port D pins to enable TXD2 and RXD2. Set PPARD[27,28] and PDIRD[27] and clear PDIRD[28] and PSORD[27,28].
2. Configure ports C and D pins to enable $\overline{\text{RTS2}}$, $\overline{\text{CTS2}}$ and $\overline{\text{CD2}}$. Set PPARD[26], PPARC[12,13] and PDIRD[26] and clear PDIRC[12,13], PSORC[12,13] and PSORD[26].
3. Configure port C pin 29 to enable the CLK3 pin. Set PPARC[29] and clear PDIRC[29] and PSORC[29].
4. Connect CLK3 to SCC2 using the CPM mux. Program CMXSCR[R2CS] and CMXSCR[T2CS] to 0b110.
5. Connect the SCC2 to the NMSI and clear CMXSCR[SC2].
6. Write RBASE with 0x0000 and TBASE with 0x0008 in the SCC2 parameter RAM to point to one RxBD at the beginning of dual-port RAM followed by one TxBD.
7. Write 0x04A1_0000 to the CPCR to execute INIT RX AND TX PARAMETERS for SCC2.
8. Write 0x0041 to the CPCR to execute INIT RX AND TX PARAMETERS for SCC2.
9. Write RFCR and TFCR with 0x10 for normal operation.
10. Write MRBLR with the maximum number of bytes per receive buffer and assume 16-bytes, so MRBLR = 0x0010.
11. Write CRC_P with 0x0000_FFFF to comply with the 16-bit CRC-CCITT.
12. Write CRC_C with 0x0000_F0B8 to comply with the 16-bit CRC-CCITT.

13. Initialize the RxBD. Assume the Rx buffer is at 0x0000_1000 in main memory. Write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x0000_1000 to RxBD[Buffer Pointer].
14. Initialize the TxBD. Assume the Tx buffer is at 0x0000_2000 in main memory and contains five 8-bit characters. Write 0xBC00 to TxBD[Status and Control], 0x0005 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer].
15. Write 0xFFFF to SCCE to clear any previous events.
16. Write 0x0013 to SCCM to enable the TXE, TXB, and RXB interrupts.
17. Write 0x0040_0000 to the SIU interrupt mask register low (SIMR_L) so SCC2 can generate a system interrupt. Initialize SIU interrupt pending register low (SIPNR_L) by writing 0xFFFF_FFFF to it.
18. Write 0x0000_1980 to GSMR_H2 to configure the transparent channel.
19. Write 0x0000_0000 to GSMR_L2 to configure $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ to automatically control transmission and reception (DIAG bits). Normal operation of the transmit clock is used. Note that the transmitter (ENT) and receiver (ENR) are not enabled yet.
20. Write 0x0000_0030 to GSMR_L2 to enable the SCC2 transmitter and receiver. This additional write ensures that the ENT and ENR bits are enabled last.

Note that after 5 bytes are sent, the Tx buffer is closed and after 16 bytes are received the Rx buffer is closed. Any data received after 16 bytes causes a busy (out-of-buffers) condition since only one RxBD is prepared.

Chapter 32
SCC AppleTalk Mode

AppleTalk is a set of protocols developed by Apple Computer, Inc. to provide a LAN service between Macintosh computers and printers. Although AppleTalk can be implemented over a variety of physical and link layers, including Ethernet, AppleTalk protocols have been most closely associated with the LocalTalk physical and link-layer protocol, an HDLC-based protocol that runs at 230.4 kbps. In this manual, the term 'AppleTalk controller' refers to the support that the MPC8560 provides for LocalTalk protocol. The AppleTalk controller provides required frame synchronization, bit sequence, preamble, and postamble onto standard HDLC frames. These capabilities, with the use of the HDLC controller in conjunction with DPLL operation in FM0 mode, provide the proper connection formats to the LocalTalk bus.

32.1 Operating the LocalTalk Bus

A LocalTalk frame, shown in Figure 32-1, is basically a modified HDLC frame.

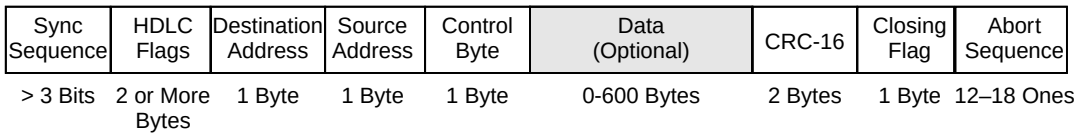


Figure 32-1. LocalTalk Frame Format

First, a synchronization sequence of more than three bits is sent. This sequence consists of at least one logical one bit (FM0 encoded) followed by two bit times or more of line idle with no particular maximum time specified. The idle time allows LocalTalk equipment to sense a carrier by detecting a missing clock on the line. The remainder of the frame is a typical half-duplex HDLC frame. Two or more flags are sent, allowing bit, byte, and frame delineation or detection. Two bytes of address, destination, and source are sent next, followed by a byte of control and 0–600 data bytes. Next, two bytes of CRC (the common 16-bit CRC-CCITT polynomial referenced in the HDLC standard protocol) are sent. The LocalTalk frame is then terminated by a flag and a restricted HDLC abort sequence. Then the transmitter's driver is disabled.

The control byte within the LocalTalk frame indicates the type of frame. Control byte values from 0x01–0x7F are data frames; control byte values from 0x80–0xFF are control frames. Four control frames are defined:

- ENQ—Enquiry
- ACK—Enquiry acknowledgment
- RTS—Request to send a data frame
- CTS—Clear to send a data frame

Frames are sent in groups known as dialogs, which are handled by the software. For instance, to transfer a data frame, three frames are sent over the network. An RTS frame (not to be confused with the RS-232 $\overline{\text{RTS}}$ pin) is sent to request the network, a CTS frame is sent by the destination node, and the data frame is sent by the requesting node. These three frames comprise one possible type of dialog. After a dialog begins, other nodes cannot start sending until the dialog is complete. Frames within a dialog are sent with a maximum interframe gap (IFG) of 200 µs. Although the LocalTalk specification does not state it, there is also a minimum recommended IFG of 50 µs. Dialogs must be separated by a minimum interdialog gap (IDG) of 400 µs. In general, these gaps are implemented by the software.

Depending on the protocol, collisions should be encountered only during RTS and ENQ frames. Once frame transmission begins, it is fully sent, regardless of whether it collides with another frame. ENQ frames are infrequent and are sent only when a node powers up and enters the network. A higher-level protocol controls the uniqueness and transmission of ENQ frames.

In addition to the frame fields, LocalTalk requires that the frame be FM0 (differential Manchester space) encoded, which requires one level transition on every bit boundary. If the value to be encoded is a logical zero, FM0 requires a second transition in the middle of the bit time. The purpose of FM0 encoding is to avoid having to transmit clocking information on a separate wire. With FM0, the clocking information is present whenever valid data is present.

32.2 Features

The following list summarizes the features of the SCC in AppleTalk mode:

- Superset of the HDLC controller features
- FM0 encoding/decoding
- Programmable transmission of sync sequence
- Automatic postamble transmission
- Reception of sync sequence does not cause extra SCCE[DCC] interrupts
- Reception is automatically disabled while sending a frame
- Transmit-on-demand feature expedites frames

- Connects directly to an RS-422 transceiver

32.3 Connecting to AppleTalk

As shown in Figure 32-2, the MPC8560 connects to LocalTalk, and, using TXD, $\overline{\text{RTS}}$, and RXD, is an interface for the RS-422 transceiver. The RS-422, in turn, is an interface for the LocalTalk connector. Although it is not shown, a passive RC circuit is recommended between the transceiver and connector.

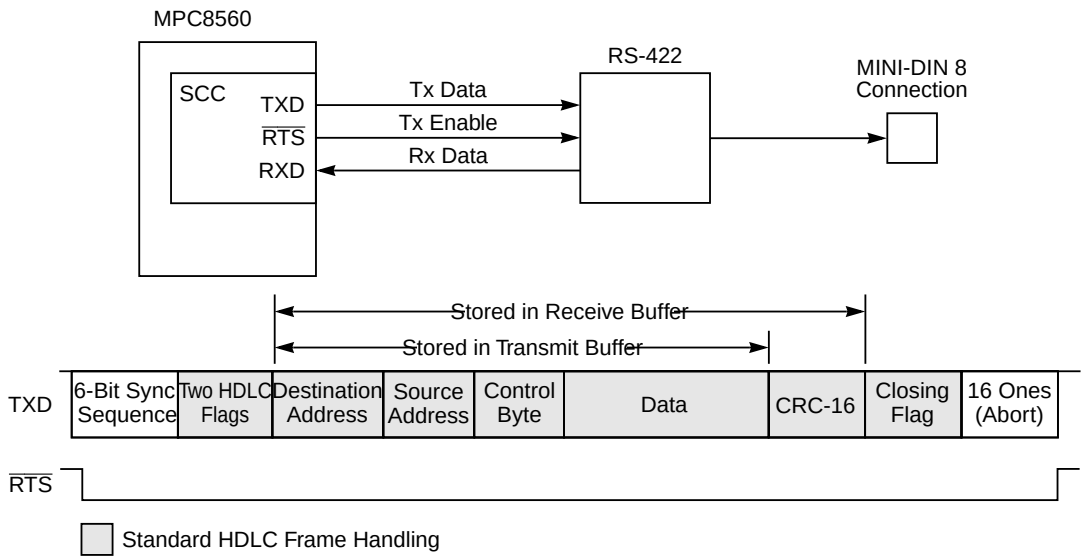


Figure 32-2. Connecting the MPC8560 to LocalTalk

The 16x overspeed of a 3.686-MHz clock can be generated from an external frequency source or from one of the baud rate generators if the resulting output frequency is close to a multiple of the 3.686 MHz frequency. The MPC8560 asserts $\overline{\text{RTS}}$ throughout the duration of the frame so that $\overline{\text{RTS}}$ can be used to enable the RS-422 transmit driver.

32.4 Programming the SCC in AppleTalk Mode

The AppleTalk controller is implemented by setting certain bits in the HDLC controller. Otherwise, Chapter 29, “SCC HDLC Mode,” describes how to program the HDLC controller. Use GSMR, PSMR, or TODR to program the AppleTalk controller.

32.4.1 Programming the GSMR

Program the GSMR as described below:

1. Set MODE to 0b0010 (AppleTalk).

2. Set DIAG to 0b00 for normal operation, with $\overline{CD}$ and $\overline{CTS}$ grounded or configured for parallel I/O. This causes $\overline{CD}$ and $\overline{CTS}$ to be internally asserted to the SCC.
3. Set RDCR and TDCR to (0b10) a 16x clock.
4. Set the TENC and RENC bits to 0b010 (FM0).
5. Clear TEND for default operation.
6. Set TPP to 0b11 for a preamble pattern of all ones.
7. Set TPL to 0b000 to transmit the next frame with no synchronization sequence and to 001 to transmit the next frame with the LocalTalk synchronization sequence. For example, data frames do not require a preceding synchronization sequence. These bits may be modified on-the-fly if the AppleTalk protocol is selected.
8. Clear TINV and RINV so data will not be inverted.
9. Set TSNC to 1.5 bit times (0b10).
10. Clear EDGE. Both the positive and negative edges are used to change the sample point (default).
11. Clear RTSM (default).
12. Set all other bits to zero or default.
13. Set ENT and ENR as the last step to begin operation.

32.4.2 Programming the PSMR

Follow these steps to program the protocol-specific mode register:

1. Set NOF to 0b0001 giving two flags before frames (one opening flag, plus one additional flag).
2. Set CRC 16-bit CRC-CCITT.
3. Set DRT.
4. Set all other bits to zero or default.

For the PSMR definition, see Section 29.8, “HDLC Mode Register (PSMR).”

32.4.3 Programming the TODR

Use the transmit-on-demand (TODR) register to expedite a transmit frame. See Section 27.2.3, “Transmit-on-Demand Register (TODR).”

32.4.4 SCC AppleTalk Programming Example

Except for the previously discussed register programming, use the example in Section 29.14.6, “HDLC Bus Protocol Programming.”

Chapter 33
Multi-Channel Controllers (MCCs)

The MPC8560's two multi-channel controllers (MCC1 and MCC2) each handle up to 128 serial, full-duplex data channels. The 128 channels are divided into 4 subgroups (of 32 channels each). One or more subgroups can be multiplexed through corresponding SI*x* TDM channels; MCC1 connects through SI1, and MCC2 uses SI2.

Each channel can be programmed separately either to perform HDLC formatting/deformatting or to act as a transparent channel.

33.1 Features

Each MCC has the following features:

- Up to 128 independent communication channels.
- Independent mapping for receive/transmit.
- Supports either transparent or HDLC protocols for each channel.
- Up to 256 DMA channels with independent buffer descriptor (BD) tables.
- Five interrupt circular tables with programmable size and overflow identification. One for transmit and four for receive.
- Global loop mode.
- Individual channel loop mode.
- Efficient bus usage (no bus usage for inactive channel or for active channels with nothing to transmit).
- Efficient control of the interrupts to the core.
- Supports external BD tables.
- Uses on-chip dual-port RAM (DPR) for parameter storage.
- Uses 64-bit data transactions for reading and writing data in BDs.
- Supports automatic routing in transparent mode using negative empty polarity.
- Supports inverted data per channel.
- Supports super channel synchronization in transparent mode (slot synchronization).

- Supports in-line synchronization in transparent mode (synchronization on a pattern of 2 bytes).

33.2 SS7 Controller

Based on the HDLC protocol, the signalling system #7 (SS7) protocol is used to manage public switching networks. The SS7 protocol operates on signal units (SU), which are analogous to HDLC frames. The physical, data link, and network layer functions of the SS7 protocol are called the message transfer part (MTP). Implementing the MTP layer 2 (data link) functions in host software is difficult with multiple performance issues. The SS7 microcode on the MPC8560 enables applications requiring multi-channel SS7 processing.

The SS7 controller is implemented using the MCC hardware with microcode running on the CPM. Each MCC implements the following layer 2 portions of the MTP:

- Signal unit (SU) retransmission
- Automatic fill-in signal unit (FISU) transmission
- Short SU filtering
- Duplicate fill-in and link-status signal unit (FISU/LSSU) filtering
- Octet counting
- Signal unit error rate monitoring
- Good frame counter and bad frame counting
- Initial alignment (Supports Alignment Error Rate Monitoring)

Host software, however, is needed to handle the following higher-level functions of the MTP layer 2 not supported by the SS7 controller:

- Link state control
- Flow control

33.2.1 SS7 Controller Features

- Up to 128 independent communication channels (64 channels per MCC)
- Independent mapping for receive and transmit
- Standard HDLC features
  - — Flag/Abort/Idle generation/detection
  - — Zero insertion/deletion
  - — 16-bit CRC-CCITT generation/checking
  - — Detection of non-octet aligned signal units
  - — Programmable number of flags between signal units
- Maintenance of signal unit error monitor (SUERM)

- Maintenance of alignment error rate monitor (AERM)
- Maintenance of separate counters for error-free and bad frames
- Detection and stripping of long signal units
- Discard of short signal units (less than 5 octets)
- Transmission of signal units with a programmable delay (applies to JT-Q.703 standard)
- Automatic transmission of fill-in signal units (FISU)
- Automatic retransmission of signal units (for link-status signal unit (LSSU) retransmission)
- Automatic discard of identical FISUs and LSSUs using a user-defined mask
- Octet counting mode in case of long signal units and receiver overrun
- Five circular interrupt tables with programmable size and overflow identification—one for transmit and four for receive.
- Global or individual channel loop modes
- Efficient bus usage (no bus usage for inactive channels or for active channels with nothing to send)
- Efficient control of interrupts to the CPU
- Supports external BD tables
- Uses on-chip dual-port RAM for parameter storage
- Uses 64-bit data transactions for reading and writing data in BDs

33.3 MCC Data Structure Organization

Each MCC uses the following data structures:

- Global MCC parameters (common to all the 128 channels) placed in the DPR from the offset (relative to the DPR base address) defined in Table 20-12..
- Channel-specific parameters. Each channel use 64 bytes of specific parameters placed in the DPR at offset 64 × CH_NUM (relative to the DPR base address). CH_NUM is the channel number (0–127 for MCC1 and 128–255 for MCC2).
- Channel-specific parameters are described in Section 33.7, “Channel-Specific HDLC Parameters,” and Section 33.8, “Channel-Specific Transparent Parameters.”
- Note that the DPR memory corresponding to the inactive channels can be used for other purposes.
- Channel extra parameters. Each channel use 8 bytes of extra parameters placed in the DPR at offset XTRABASE + 8 × CH_NUM (relative to the DPR base address). XTRABASE is one of the global MCC parameters.

- Channel extra parameters are described in Section 33.5, “Channel Extra Parameters.” Note that the DPR memory corresponding to the inactive channels can be used for other purposes.
- Super channel table (used only if super channels are defined). This table is placed in the DPR from the offset SCTPBASE (relative to the DPR base address). SCTPBASE is one of the global MCC parameters. The super channel table is described in Section 33.6, “Super-Channel Table.”
- BD tables placed in the external memory. All the BD tables associated with one MCC must reside in a 512-KByte segment. The absolute base addresses of a channel BD table is MCCBASE + 8 × RBASE (for the receiver) and MCCBASE + 8 × TBASE (for the transmitter). MCCBASE is one of the global MCC parameters and RBASE/TBASE are channel extra parameters. Each BD table is a circular queue. One BD includes status bits, start address and length of a data buffer. Figure 33-1 shows the BD structure for one MCC.
- Circular interrupt tables placed in the external memory. There is one table for the transmitter interrupts (base address TINTBASE) and between one and four tables for receiver interrupts (base address RINTBASE0–RINTBASE4). TINTBASE and RINTBASE0–RINTBASE4 are global MCC parameters.
- Three registers (MCCE, MCCM, and MCCF) at described in Section 33.12.1, “MCC Event Register (MCCE)/Mask Register (MCCM),” and Section 33.10, “MCC Configuration Registers (MCCFx).”

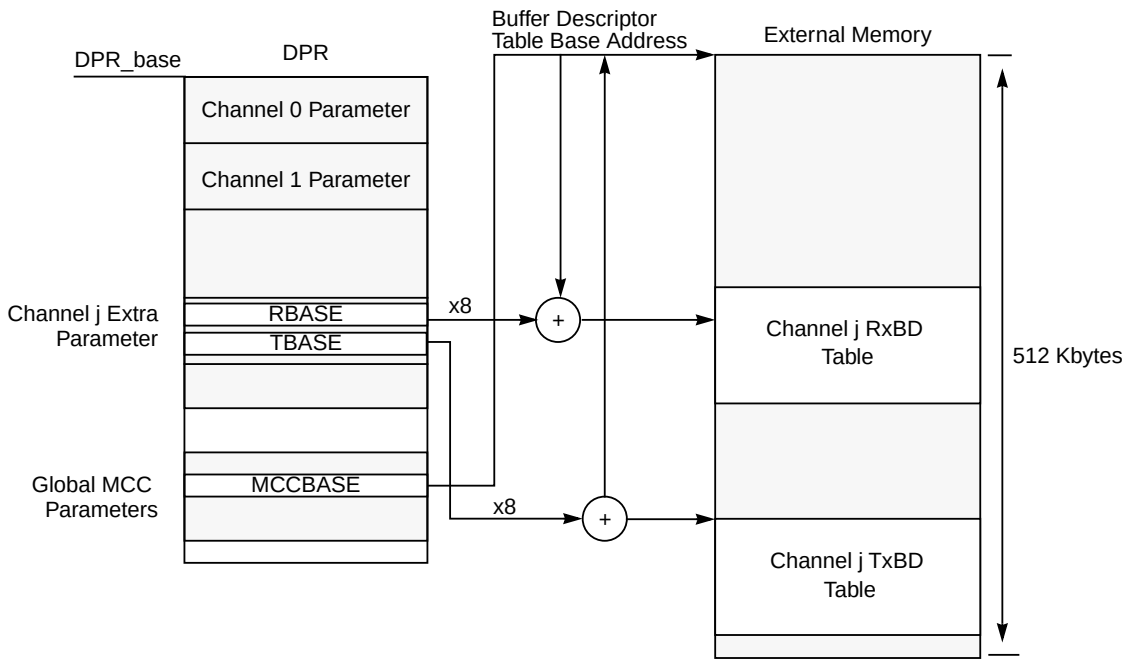


Figure 33-1. BD Structure for One MCC

33.4 Global MCC Parameters

The global MCC parameters are described in Table 33-1.

Table 33-1. Global MCC Parameters

Offset [1]	Name	Width	Description
0x00	MCCBASE	Word	MCC base pointer. User-initialized parameter points to the starting address of a 512-Kbyte BD segment in external memory.
0x04	MCCSTATE	Hword	MCC state, used by the CP for global state definition. Should be cleared during initialization.
0x06	MRBLR	Hword	Maximum receive buffer length (user-initialized). Defines the maximum number of bytes written to a receive buffer before moving to the next buffer for this channel. This value must be a multiple of 8.
0x08	GRFTHR	Hword	Global receive frame threshold. Used to reduce interrupt overhead that can occur when many short HDLC frames arrive that each cause an RXF interrupt. Programming GRFTHR can limit the frequency of RXF interrupts. Note that an RXF event is written to the interrupt queue on each received frame but GINT is set only when the number of RXF events (by all channels) reaches the GRFTHR value. This parameter does not need to be reset after an interrupt.
0x0A	GRFCNT	Hword	Global receive frame count. A down counter used to implement the GRFTHR feature. It should be initialized to the GRFTHR value. The CP writes an entry in a circular interrupt table and decrements GRFCNT each time a frame is received. When GRFCNT reaches zero, the CP generates an interrupt and re-initializes GRFCNT with GRFTHR. This parameter does not need to be reset after an interrupt.
0x0C	RINTTMP	Word	Temporary location for holding the receive interrupt queue entry, used by the CP (reserved)
0x10	DATA0	Word	Temporary location for holding data, used by the CP (reserved)
0x14	DATA1	Word	Temporary location for holding data, used by the CP (reserved)
0x18	TINTBASE	Word	Multi-channel transmitter circular interrupt table base address. The interrupt circular table is a cyclic table (FIFO-like). Each table entry contains information about an interrupt request generated by the MCC to the host.
0x1C	TINTPTR	Word	Pointer to the transmitter circular interrupt table. The CP writes the next interrupt information to this entry when an exception occurs. The user must copy the TINTBASE value to TINTPTR before enabling interrupts. Further updates of the TINTPTR are done by the CP.
0x20	TINTTMP	Word	Temporary location for holding the transmit interrupt queue entry, used by the CP. The CPU initializes this field before initializing the MCC. The user must clear it before enabling interrupts.
0x24	SCTPBASE	Hword	Internal pointer for the super channel transmit table, offset from the DPRAM address
0x26	—	Hword	
0x28	C_MASK32	Word	CRC constant (user initialized to 0xDEBB20E3). Used for 32-bit CRC-CCITT calculation if HDLC mode is chosen for a selected channel. (This option is programmable. For each HDLC channel, one of two CRC-CCITT can be selected through the CHAMR.)
0x2C	XTRABASE	Hword	Pointer the beginning of the extra parameters information, offset from the DPRAM address

Table 33-1. Global MCC Parameters (continued)

Offset [1]	Name	Width	Description
0x2E	C_MASK16	Hword	CRC constant (user initialized to 0xF0B8). Used for 16-bit CRC-CCITT calculation if HDLC mode is chosen for a selected channel. This option is programmable. For each HDLC channel, one of two CRC-CCITT can be selected through the CHAMR.
0x30	RINTTMP0	Word	RINTTMPx. Temporary location for holding a receive circular interrupt table entry (for tables 0–4), used by the CP. The user must clear it before enabling interrupts.See Section 33.12, “MCC Exceptions.”
0x34	RINTTMP1	Word	
0x38	RINTTMP2	Word	
0x3C	RINTTMP3	Word	
0x40	RINTBASE0	Word	RINTBASEx—Multi-channel receiver circular interrupt table base address. The interrupt circular table is a cyclic table (FIFO-like). Each table entry contains information about an interrupt request generated by the MCC to the host. RINTPTRx—Pointer to the receiver circular interrupt table. The CP writes the next interrupt information to this entry when an exception occurs. The user must copy the RINTBASEx value to RINTPTRx before enabling interrupts. Further updates of the RINTPTRx are done by the CP.
0x44	RINTPTR0	Word	
0x48	RINTBASE1	Word	
0x4C	RINTPTR1	Word	
0x50	RINTBASE2	Word	
0x54	RINTPTR2	Word	
0x58	RINTBASE3	Word	
0x5C	RINTPTR3	Word	
0x60	TS_TMP	Word	Temporary place for time stamp

1 Offset to MCC Base

33.5 Channel Extra Parameters

Table 33-2 describes extra parameters. This table is indexed by logical channel number.

Table 33-2. Channel Extra Parameters

Offset [1]	Name	Width	Description
0x00	TBASE	Hword	TxBD base address. Offset of the channel’s TxBD table relative to the MCCBASE (The base address of the BD table for this channel MCCBASE+8 × TBASE)
0x02	TBPTR	Hword	TxBD pointer. Offset of the current BD relative to the MCCBASE. TBPTR is user-initialized to TBASE before enabling the channel or after a fatal error before reinitializing the channel. (The address of the BD in use for this channel MCCBASE+8 × TBPTR)
0x04	RBASE	Hword	RxBD base address. Offset of the channel’s RxBD table relative to the MCCBASE. (The base address of the BD table for this channel MCCBASE+8 × RBASE)
0x06	RBPTR	Hword	RxBD pointer. Offset of the current BD relative to the MCCBASE. RBPTR is user-initialized to RBASE before enabling the channel or after a fatal error before reinitializing the channel. (The address of the BD in use for this channel MCCBASE+8 × RTBPTR)

1 The offset relative to dual-port RAM base address + XTRABASE + 8 × CH_NUM

33.6 Super-Channel Table

The super channel table entry redirects an MCC slot to a different channel number. For this reason, the transmitter super channel uses more FIFO (2 bytes—half of a single channel transmitter FIFO—multiplied by the number of the channels in the super channel) in the MCC hardware.

On the transmitter side, super channels must be defined in the SI RAM (see Section 22.4.3, “Programming SIx RAM Entries,” for details) and a super-channel table must be created.

On the receiver side, the transparent super channels that require slot synchronization must be programmed in the SI RAM as super channels (the slot synchronization ensures that the data is aligned in the receiver buffer starting from the first time slot after a sync pulse). In this case, 1 byte of FIFO is allocated for each super channel. Transparent super channels that do not require slot synchronization and HDLC super channels can be programmed in the SI RAM as regular channels pointing to the same MCC channel. In this case the FIFO allocated for each super channel is 2 bytes (and the CP load will be lower).

	0	1	2 … 9	10	11	12	13	14	15
Field	0	0	Channel Number	0	0	0	0	0	0
Addr	DPR_base_address + SCTPBASE + 2 × Virtual_Channel_Number (Virtual_channel_number is the number written in the MCSEL field of the corresponding SI RAM entry)								

Figure 33-2. Super Channel Table Entry

Note that two consecutive slots (of the same super channel) must not be associated with the same virtual channel; that is, the MCSEL values should not be the same in two consecutive SI RAM entries.

The example in Figure 33-3 shows the SI RAM programming and the super-channel table for two transmitter super channels, one including slots 1, 6, and 7 and the second 2, 3, and 4. Figure 33-4 shows the SI RAM programming for the same transparent receiver super channels which uses the slot synchronization. Figure 33-5 shows the SI RAM programming for the same transparent or HDLC receiver super channels that do not use slot synchronization.

SI RAM

0	1	2	3–10	11–13	14	15
MCC	LOOP	SUPER	MCSEL	CNT	BYT	LST
SI RAM Address						
1	0	0	0x0	0x1	1	0
1	0	1	0x1	0x0 [1]	1	0
1	0	1	0x2	0x0[1]	1	0
1	0	1	0x3	0x7 [2]	0	0
1	0	1	0x4	0x7[2]	0	0
1	0	0	0x5	0x1	1	0
1	0	1	0x6	0x7[2]	0	0
1	0	1	0x7	0x7[2]	0	0
1	0	0	0x8	0x1	1	1

Super Channel Table

	0–1	2–9	10–15
		CHANNEL NO	
	DPR_Base + SCTPBASE +		
0x0		—	
0x2		0x1	
0x4		0x2	
0x6		0x2	
0x8		0x2	
0xA		—	
0xC		0x1	
0xE		0x1	
0x10		—	

[1] First slot of the super channel

[2] Regular (not first) slot of the super channel

The super channel BD tables are associated with channels 1 and 2 (no BD tables are necessary for channels 3, 4, 6, and 7)

Figure 33-3. Transmitter Super Channel Example

The example in Figure 33-4 shows a receiver super channel with slot synchronization.

SI RAM

0	1	2	3–10	11–13	14	15	
MCC	LOOP	SUPER	MCSEL	CNT	BYT	LST	
SI RAM Address							
1	0	0	0x0	0x1	1	0	Regular Channel
1	0	1	0x1	0x0 [1]	1	0	Super Channel 1
1	0	1	0x2	0x0[1]	1	0	Super Channel 2
1	0	1	0x2	0x7 [2]	0	0	Super Channel 2
1	0	1	0x2	0x7[2]	0	0	Super Channel 2
1	0	0	0x3	0x1	1	0	Regular Channel
1	0	1	0x1	0x7[2]	0	0	Super Channel 1
1	0	1	0x1	0x7[2]	0	0	Super Channel 1
1	0	0	0x1	0x1	1	1	Regular Channel

[1] First slot of the super channel

[2] Regular (not first) slot of the super channel

The super channel BD tables are associated with channels 1 and 2.

Figure 33-4. Receiver Super Channel with Slot Synchronization Example

The example in Figure 33-5 shows a receiver super channel without slot synchronization.

SI RAM

0	1	2	3–10	11–13	14	15	
MCC	LOOP	SUPER	MCSEL	CNT	BYT	LST	
SI RAM Address							
1	0	0	0x0	0x1	1	0	Regular Channel
1	0	0	0x1	0x1	1	0	Super Channel 1
1	0	0	0x2	0x1	1	0	Super Channel 2
1	0	0	0x2	0x1	1	0	Super Channel 2
1	0	0	0x2	0x1	1	0	Super Channel 2
1	0	0	0x3	0x1	1	0	Regular Channel
1	0	0	0x1	0x1	1	0	Super Channel 1
1	0	0	0x1	0x1	1	0	Super Channel 1
1	0	0	0x4	0x1	1	1	Regular Channel

The super channel BD tables are associated with channels 1 and 2.

Figure 33-5. Receiver Super Channel without Slot Synchronization Example

33.7 Channel-Specific HDLC Parameters

Table 33-3 describes channel-specific parameters for HDLC.

Table 33-3. Channel-Specific Parameters for HDLC

Offset [1]	Name	Width	Description
0x00	TSTATE	Word	Tx internal state. To start a transmitter channel the user must write to TSTATE 0xHH80_0000. HH is the TSTATE high byte described in Section 33.7.1, “Internal Transmitter State (TSTATE).”
0x04	ZISTATE	Word	Zero-insertion machine state.(User-initialized to 0x10000207 for regular channel, and 0x30000207 for inverted channel)
0x08	ZIDATA0	Word	Zero-insertion high word data buffer (User-initialized to 0xFFFFFFFF)
0x0C	ZIDATA1	Word	Zero-insertion low word data buffer (User-initialized to 0xFFFFFFFF)
0x10	TBDFlags	Hword	TxDB flags, used by the CP (read-only for the user)
0x12	TBDCNT	Hword	Tx internal byte count. Number of remaining bytes in buffer, used by the CP (read-only for the user)
0x14	TBDPTR	Word	Tx internal data pointer. Points to current absolute data address of channel, used by the CP (read-only for the user)
0x18	INTMSK	Hword	Channel’s interrupt mask flag. See Section 33.7.2, “Interrupt Mask (INTMSK).”
0x1A	CHAMR	Hword	Channel mode register. See Section 33.7.3, “Channel Mode Register (CHAMR).”
0x1C	TCRC	Word	Temp transmit CRC. Temp value of CRC calculation result, used by the CP (read-only for the user)
0x20	RSTATE	Word	Rx internal state. To start a receiver channel the user must write to RSTATE 0xHH80_0000. HH is the RSTATE high byte described in Section 33.7.4, “Internal Receiver State (RSTATE).”
0x24	ZDSTATE	Word	Zero-deletion machine state (User-initialized to 0x00FFFFE0 for regular channel and 0x20FFFFE0 for inverted channel)
0x28	ZDDATA0	Word	Zero-deletion high word data buffer (User-initialized to 0xFFFFFFFF)
0x2C	ZDDATA1	Word	Zero-deletion low word data buffer (User-initialized to 0xFFFFFFFF)
0x30	RBDFlags	Hword	RxBD flags, used by the CP (read-only for the user)
0x32	RBDCNT	Hword	Rx internal byte count. Number of remaining bytes in buffer, used by the CP (read-only for the user)
0x34	RBDPTR	Word	Rx internal data pointer. Points to current absolute data address of channel, used by the CP (read-only for the user)
0x38	MFLR	Hword	Maximum frame length register. Defines the longest expectable frame for this channel. (64-Kbyte maximum). The remainder of a frame that is larger than MFLR is discarded and the LG flag is set in the last frame’s BD. An interrupt request might be generated (RXF and RXB) depending on the interrupt mask. A frame’s length is considered to be everything between flags, including CRC. No more data is written into the current buffer when the MFLR violation is detected.
0x3A	MAX_CNT	Hword	Max_length counter, used by the CP (read-only for the user)
0x3C	RCRC	Word	Temp receive CRC, used by the CP (read-only for the user)

[1] The offset is relative to dual-port RAM base address + 64 × CH_NUM

33.7.1 Internal Transmitter State (TSTATE)

Internal transmitter state (TSTATE) is a 4-byte register provides transaction parameters associated with SDMA channel accesses (like function code registers) and starts the transmitter channel.

To start the channel, write 0xHH800000 to TSTATE, where HH is the TSTATE high byte (see Figure 33-6). When the channel is active, the CP changes the value of the three LSBs, hence these 3 bytes must be masked if the user reads back the TSTATE.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		—	DTB	BDB
Reset	—							
R/W	R/W							

Figure 33-6. TSTATE High Byte

TSTATE high-byte fields are described in Table 33-4.

Table 33-4. TSTATE High-Byte Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global. Setting GLB activates snooping (only the system bus can be snooped, this parameter is ignored for local bus transactions).
3–4	BO	Byte ordering. 0x Reserved 1x Big-endian
5	—	Reserved, should be cleared.
6	DTB	Data bus indicator. Selects the bus that handles transfers to and from data buffers. 0 System bus SDMA 1 Local bus SDMA
7	BDB	BD bus. Sets the bus that handles transfers to/from BD and interrupt circular tables. 0 System bus SDMA used for accessing BDs 1 Local bus SDMA used for accessing BDs

33.7.2 Interrupt Mask (INTMSK)

The interrupt mask (INTMSK), shown in Figure 33-7, provides in bits for enabling/disabling each event defined in the interrupt circular table entry.

	0	5	6	7	8	9	10	11	12	13	14	15
Interrupt Entry	—		UN	TXB	—		NID	IDL	MRF	RXF	BSY	RXB
INTMSK	—		Mask Bits		—		Mask Bits					

Figure 33-7. INTMSK Mask Bits

To enable an interrupt, set the corresponding bit. If a bit is cleared, no interrupt request is generated and no new entry is written in the circular interrupt table. The user must initialize INTMSK prior to operation. Reserved bits are cleared.

33.7.3 Channel Mode Register (CHAMR)

The channel mode register (CHAMR) is a user-initialized register, shown in Figure 33-8. This is a generalized representation of CHAMR. Section 33.8.1, “Channel Mode Register (CHAMR)—Transparent Mode,” describes the CHAMR for transparent mode.

	0	1	2	3	4	5	6 7	8	9	10	11 12	13 15
Field	MODE	POL	1	IDLM	—	RD	—	CRC	—	TS	RQN	NOF
Reset	—											
R/W	R/W											
Offset	0x1A											

Figure 33-8. Channel Mode Register (CHAMR)

CHAMR fields are described in Figure 33-5.

Table 33-5. CHAMR Field Descriptions

Bits	Name	Description
0	MODE	This mode bit determines whether the HDLC or transparent mode is used. It also determines how other CHAMR bits are interpreted. 0 Transparent mode. See Section 33.8.1, “Channel Mode Register (CHAMR)—Transparent Mode.” 1 HDLC mode
1	POL	Enable polling. POL enables the transmitter to poll the TxBDs. 0 Polling is disabled (The CPM does not access the external bus to check the R bit in the TxBD). 1 Polling is enabled. POL can be used to optimize the use of the external bus. Software should always set POL at the beginning of a transmit sequence of one or more frames. The CP clears POL when no more buffers are ready in the transmit queue, i.e. when it finds a BD with R = 0 (for example, at the end of a frame or at the end of a multi-frame transmission). To minimize useless transactions on the external bus, software should always prepare the new BD, or multiple BDs, and set BD[R] before enabling polling.
2	1	Must be set.
3	IDLM	Idle mode. 0 No idle patterns are sent between frames. After sending NOF+1 flags, the transmitter starts sending the data of the frame. If the transmission is between frames and the frame buffers are not ready, the transmitter sends flags until it can start transmitting the data. 1 At least one idle pattern is sent between adjacent frames. The NOF value shall be no smaller than the PAD setting, see TxBD. If NOF = 0, this is identical to flag sharing in HDLC. Mode flags precede the actual data. When IDLM = 1, at least one idle pattern is sent between adjacent frames. If the transmission is between frames and the frame buffer is not ready, the transmitter sends idle characters. When data is ready, the NOF + 1 flags are sent followed by the data frame. If IDLE mode is selected and NOF = 1, the following sequence is sent: .....init value, FF, FF, flag, flag, data, ... The init value before the idle will be ones.
4	—	These bits must be cleared.

Table 33-5. CHAMR Field Descriptions (continued)

Bits	Name	Description
5	RD	0 Normal bit order (transmit/receive the lsb of each octet first) 1 Reversed bit order (transmit/receive the msb of each octet first).
6–7	—	These bits must be cleared.
8	CRC	Selects the type of CRC when HDLC channel mode is used. 0 16-bit CCITT-CRC 1 32-bit CCITT-CRC
9	—	This bit must be cleared.
10	TS	Receive time stamp. If this bit is set a 4 byte time stamp is written at the beginning of every data buffer that the BD points to.If this bit is set the data buffer must start from an address equal to 8*n-4 (n is any integer larger than 0).
11–12	RQN	Receive queue number. Specifies the receive interrupt queue number. 00 Queue number 0. 01 Queue number 1. 10 Queue number 2. 11 Queue number 3.
13–15	NOF	Number of flags. NOF defines the minimum number of flags before frames: 000 At least 1 flag 001 At least 2 flags .... 111 At least 8 flags

33.7.4 Internal Receiver State (RSTATE)

Internal receiver state (RSTATE) is a 4-byte register that provides transaction parameters associated with SDMA channel accesses (like function code registers) and starts the receiver channel.

To start the channel the user must write 0xHH800000 to RSTATE, where HH is the RSTATE high byte (see Figure 33-9). When the channel is active the CP changes the value of the 3 LSBs, hence these 3 bytes must be masked if the user reads back the RSTATE.

	0	1	2	3	4	5	6	7
Field	—		GBL	BO		—	DTB	BDB
Reset	—							
R/W	R/W							
Addr	0x20							

Figure 33-9. Rx Internal State (RSTATE) High Byte

RSTATE high-byte fields are described in Table 33-6.

Table 33-6. RSTATE High-Byte Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global. Setting GLB activates snooping (only the system bus can be snooped, this parameter is ignored for local bus transactions).
3–4	BO	Byte ordering. 0x Reserved 1x Big-endian
5	—	Reserved, should be cleared.
6	DTB	Data bus indicator. The transfers to data buffers are handled by the: 0 system bus SDMA 1 Local bus SDMA
7	BDB	BD and interrupt circular tables bus indicator. The transfers to/from BD and interrupt circular tables are handled by the: 0 system bus SDMA 1 Local bus SDMA Note that the following restrictions result from the fact that there is a common bus selection bit for BDs and interrupt circular tables: • The RxBDs of all the channels that use a particular interrupt table must reside on the same bus (system or local). • All TxBDs must reside on the same bus (system or local).

33.8 Channel-Specific Transparent Parameters

Table 33-7 describes channel-specific parameters for transparent operation.

Table 33-7. Channel-Specific Parameters for Transparent Operation

Offset [1]	Name	Width	Description
0x00	TSTATE	Word	Tx internal state. To start a transmitter channel the user must write to TSTATE 0xHH80_0000. HH is the TSTATE high byte described in Section 33.7.1, “Internal Transmitter State (TSTATE).”
0x04	ZISTATE	Word	Zero-insertion machine state.(User-initialized to 0x10000207 for regular channel, and 0x30000207 for inverted channel)
0x08	ZIDATA0	Word	Zero-insertion high word data buffer (User-initialized to 0xFFFFFFFF)
0x0C	ZIDATA1	Word	Zero-insertion low word data buffer (User-initialized to 0xFFFFFFFF)
0x10	TBDFlags	Hword	TxDB flags, used by the CP (read-only for the user)
0x12	TBDCNT	Hword	Tx internal byte count. Number of remaining bytes in buffer, used by the CP (read-only for the user)
0x14	TBDPTR	Word	Tx internal data pointer. Points to current absolute data address of channel, used by the CP (read-only for the user)
0x18	INTMSK	Hword	Channel’s interrupt mask flag. See Section 33.7.2, “Interrupt Mask (INTMSK).”
0x1A	CHAMR	Hword	Channel mode register. See Section 33.8.1, “Channel Mode Register (CHAMR)—Transparent Mode.”

Table 33-7. Channel-Specific Parameters for Transparent Operation (continued)

<table>
<tr><th>Offset [1]</th><th>Name</th><th>Width</th><th>Description</th></tr>
<tr><td>0x1C</td><td>—</td><td>Word</td><td>Reserved</td></tr>
<tr><td>0x20</td><td>RSTATE</td><td>Word</td><td>Rx internal state. To start a receiver channel the user must write to RSTATE 0xHH80_0000. HH is the RSTATE high byte described in Section 33.7.4, “Internal Receiver State (RSTATE).”</td></tr>
<tr><td>0x24</td><td>ZDSTATE</td><td>Word</td><td>Zero-deletion machine state. Initialize ZDSTATE as in the following table:
<table>
<tr><th>Channel Type</th><th>RCVSYNC</th><th>ZDSTATE Initial Programmed Value</th></tr>
<tr><td colspan="3">If pattern synchronization is used (CHAMR[SYNC] = 1x), then…</td></tr>
<tr><td rowspan="2">Regular</td><td>0bxxxx_xxxx_xxxx_xxx0</td><td>0x00FF_FFE0</td></tr>
<tr><td>0bxxxx_xxxx_xxxx_xxx1</td><td>0x0000_0000</td></tr>
<tr><td rowspan="2">Inverted</td><td>0bxxxx_xxxx_xxxx_xxx0</td><td>0x20FF_FFE0</td></tr>
<tr><td>0bxxxx_xxxx_xxxx_xxx1</td><td>0x2000_0000</td></tr>
<tr><td colspan="3">If pattern synchronization is not used (CHAMR[SYNC] = 00), then…</td></tr>
<tr><td>Regular</td><td>0x0000</td><td>0x50FF_FFE0</td></tr>
<tr><td>Inverted</td><td>0x0000</td><td>0x70FF_FFE0</td></tr>
</table></td></tr>
<tr><td>0x28</td><td>ZDDATA0</td><td>Word</td><td>Zero-deletion high word data buffer (User-initialized to 0xFFFFFFFF)</td></tr>
<tr><td>0x2C</td><td>ZDDATA1</td><td>Word</td><td>Zero-deletion low word data buffer (User-initialized to 0xFFFFFFFF)</td></tr>
<tr><td>0x30</td><td>RBDFlags</td><td>Hword</td><td>RxBD flags, used by the CP (read-only for the user)</td></tr>
<tr><td>0x32</td><td>RBDCNT</td><td>Hword</td><td>Rx internal byte count. Number of remaining bytes in buffer, used by the CP (read-only for the user)</td></tr>
<tr><td>0x34</td><td>RBDPTR</td><td>Word</td><td>Rx internal data pointer. Points to current absolute data address of channel, used by the CP (read-only for the user)</td></tr>
<tr><td>0x38</td><td>TMRBLR</td><td>Hword</td><td>Transparent maximum receive buffer length. Defines the maximum number of bytes written to a receiver buffer before moving to the next buffer for the respective channel. This value must be 8 byte aligned.</td></tr>
<tr><td>0x3A</td><td>RCVSYNC</td><td>Hword</td><td>Receive synchronization pattern. Defines the synchronization pattern when CHAMR[SYNC] is 0b1x. The two bytes are checked in reverse order (byte from address 0x3B first and byte from address 0x3A last). Non-inverted data is used for synchronization even if the channel is programmed to invert the data. Clear RCVSYNC when CHAMR[SYNC] = 0b0x.</td></tr>
<tr><td>0x3C</td><td>—</td><td>Word</td><td>Reserved</td></tr>
</table>

[1] The offset is relative to dual-port RAM address 64*CH_NUM

33.8.1 Channel Mode Register (CHAMR)—Transparent Mode

Figure 33-10 shows the user-initialized channel mode register, CHAMR, for transparent mode.

	0	1	2	3	4	5	6 7	8 9	10	11 12	13 15
Field	MODE	POL	1	1	EP	RD	SYNC	—	TS	RQN	—
Reset	—										
R/W	R/W										
Offset	0x1A										

Figure 33-10. Channel Mode Register (CHAMR)—Transparent Mode

CHAMR fields are described in Figure 33-8,

Table 33-8. CHAMR Field Descriptions—Transparent Mode

Bits	Name	Description			
0	MODE	Channel mode. Selects either HDLC or transparent mode. 0 Transparent mode. 1 HDLC mode			
1	POL	Enable polling. POL enables the transmitter to poll the TxBDs. 0 Polling is disabled (The CPM does not access the external bus to check the R bit in the TxBD). 1 Polling is enabled. POL can be used to optimize the use of the external bus. Software should always set POL at the beginning of a transmit sequence of one or more frames. The CP clears POL when no more buffers are ready in the transmit queue, i.e. when it finds a BD with R = 0 (for example, at the end of a frame or at the end of a multi-frame transmission). To prevent a significant number of useless transactions on the external bus, software should always prepare the new BD, or multiple BDs, and set BD[R] before enabling polling.			
2–3	0b11	Must be set.			
4	EP	Empty polarity and enable polling. 0 The E bit in the RxBD is handled in positive logic (1 = empty; 0 = not empty). Polling occurs only if POL is set. 1 The E bit in the RxBD is handled in negative logic (0 = empty, 1 = not empty). Polling occurs disregarding the value of POL.			
5	RD	0 Normal bit order (transmit/receive the lsb of each octet first) 1 Reversed bit order to be reversed (transmit/receive the msb of each octet first).			
6–7	SYNC	Synchronization. SYNC controls synchronization of multi-channel operation in transparent mode.			
		SYNC	**Receive**	**Transmit**	**Description**
		00	None	None	Transmitter and receiver operate with no synchronization algorithm. RCVSYNC should be cleared or received data may be shifted.
		01	Slot	Slot	The first data is sent/received in the slot defined in the slot assignment table (for super channels only). RCVSYNC should be cleared or received data may be shifted.
		10	8-bit	None	Receive data synchronization uses an 8-bit pattern specified by the 8 msb of RCVSYNC. The sync bytes are not written to the receive buffer.
		11	16-bit	None	Receive data synchronization uses a 16-bit pattern specified by RCVSYNC. The first byte of the sync pattern will not be written to the receive buffer. The second byte of the sync pattern will be written to the receive buffer (first and second represent the order in which the two bytes of the sync pattern are received on the serial channel).

Table 33-8. CHAMR Field Descriptions—Transparent Mode (continued)

Bits	Name	Description
8–9	—	Reserved, must be cleared.
10	TS	Receive time stamp. If this bit is set a 4 byte time stamp is written at the beginning of every data buffer that the BD points to.If this bit is set the data buffer must start from an address equal to 8*N-4 (N is any number larger than 0).
11–12	RQN	Receive queue number. Specifies the receive interrupt queue number. 00 Queue number 0. 01 Queue number 1. 10 Queue number 2. 11 Queue number 3.
13–15	—	Reserved, must be cleared.

33.9 Channel-Specific SS7 Parameters

Table 33-9 describes channel-specific parameters for SS7. Note that a given parameter location may have a different definition depending on the standard used (ITU-T/ANSI or Japanese standard).

Table 33-9. Channel-Specific Parameters for SS7

Offset [1]	Name	Width	Description
0x00	TSTATE	Word	Tx internal state. The user must write to TSTATE 0xHH80_0000. HH is the TSTATE High Byte.
0x04	ZISTATE	Word	Zero-insertion machine state.(User-initialized to 0x10000207 for regular channel, and 0x30000207 for reversed bit order channel)
0x08	ZIDATA0	Word	Zero-insertion high word data buffer (User-initialized to 0xFFFFFFFF)
0x0C	ZIDATA1	Word	Zero-insertion low word data buffer (User-initialized to 0xFFFFFFFF)
0x10	TBDFlags	Hword	TxBD flags, used by the CP (read-only for the user)
0x12	TBDCNT	Hword	Tx internal byte count. Number of remaining bytes in buffer, used by the CP (read-only for the user)
0x14	TBDPTR	Word	Tx internal data pointer. Points to current absolute data address of channel, used by the CP (read-only for the user)
0x18	ECHAMR	Word	Extended channel mode register. See 33.9.1, Extended Channel Mode Register (ECHAMR).
0x1C	TCRC	Word	Temporary transmit CRC. Temporary value of CRC calculation result, used by the CP (read-only for the user)
0x20	RSTATE	Word	Rx internal state. To start a receiver channel the user must write to RSTATE 0xHH80_0000. HH is the RSTATE High Byte.
0x24	ZDSTATE	Word	Zero-deletion machine state (User-initialized to 0x00FFFFE0 for regular channel, and 0x20FFFFE0 for reversed bit order channel)
0x28	ZDDATA0	Word	Zero-deletion high word data buffer (User-initialized to 0xFFFFFFFF)
0x2C	ZDDATA1	Word	Zero-deletion low word data buffer (User-initialized to 0xFFFFFFFF)

Table 33-9. Channel-Specific Parameters for SS7 (continued)

Offset [1]	Name	Width	Description
0x30	RBDFlags	Hword	RxBD flags, used by the CP (read-only for the user)
0x32	RBDCNT	Hword	Rx internal byte count. Number of remaining bytes in buffer, used by the CP (read-only for the user)
0x34	RBDPTR	Word	Rx internal data pointer. Points to current absolute data address of channel, used by the CP (read-only for the user)
0x38	MFLR	Hword	Maximum frame length register. Defines the longest expected frame for this channel. (64-Kbyte maximum). The remainder of a frame that is larger than MFLR is discarded and the LG flag is set in the last frame's BD. An interrupt request might be generated (RXF and RXB) depending on the interrupt mask. A frame's length is considered to be everything between flags, including CRC. No more data is written into the current buffer when the MFLR violation is detected.
0x3A	MAX_cnt	Hword	Max_length counter, used by the CP (read-only for the user)
0x3C	RCRC	Word	Temporary receive CRC, used by the CP (read-only for the user)
0x40	N	Hword	Applies to ITU-T/ANSI SS7 only. Interrupt threshold in octet counting mode (N=16). See 33.9.2, Signal Unit Error Monitor (SUERM)
	N_cnt	Hword	Applies to ITU-T/ANSI SS7 only. Temporary down counter for N (user initialized to the value of N).
	JTSTTmp	Word	Applies to Japanese SS7 only. Temporary storage for Time-Stamp Register Value. Used by the CP to implement a 24-ms delay before sending FISU.
0x44	D	Hword	Signal unit to signal unit error ratio (SUERM parameter, user initialized to 256). See 33.9.2, Signal Unit Error Monitor (SUERM)
0x46	D_cnt	Hword	Applies to ITU-T/ANSI SS7 only. Temporary down-counter for D (user initialized to the value of D). D_cnt is decremented only when receive buffers are available.
	JTTDelay		Applies to Japanese SS7 only. FISU retransmission delay (specified in units of 512¨µs). According to the Japanese SS7 standard, the delay should be 24 ms and thus JTTDelay should be programmed to 24 ms/512 µs = 46.875 (approximately 47). Hence, the user should program JTTDelay to 0x2F and the RTSCR to generate a 1 µs time stamp period. Refer to Section 20.3.7, "RISC Time-Stamp Control Register (RTSCR).
0x48	Mask1	Word	Mask for SU filtering, bytes 0-3. See 33.9.4.4, SU Filtering
0x4C	Mask2	Hword	Mask for SU filtering, byte 4. See 33.9.4.4, SU Filtering
0x4E	SS7_OPT	Hword	SS7 configuration register. See 33.9.4, SS7 Configuration Register (SS7_OPT).
0x50	LRB1_Tmp	Word	Temporary storage, used by CP for SU filtering.
0x54	LRB2_Tmp	Hword	Temporary storage, used by CP for SU filtering.
0x56	SUERM	Hword	Signal unit error rate monitor counter (user initialized to 0). See Section 33.9.2, Signal Unit Error Monitor (SUERM).
0x58	LRB1	Word	Four first bytes of last received signal unit. Used by CP for SU filtering. See 33.9.4.4, SU Filtering.
0x5C	LRB2	Hword	Fifth byte of last received signal unit. Used by CP for SU filtering. See 33.9.4.4, SU Filtering
0x5E	T	Hword	SUERM threshold value (user initialized to 64). See 33.9.2, Signal Unit Error Monitor (SUERM).

Table 33-9. Channel-Specific Parameters for SS7 (continued)

Offset [1]	Name	Width	Description
0x60	LHDR	Word	The BSN, BIB, FSN, FIB fields of last transmitted signal unit and result of CRC. Used by CP for automatic FISU transmission.
0x64	LHDR_Tmp	Word	Temporary storage, used by CP for automatic FISU transmission.
0x68	EFSUC	Word	Error-free signal unit counter, user initialized to 0. The counter is incremented whenever an error-free (no CRC error, no non-octet aligned error, no short or long frame errors) signal unit is received.
0x6C	SUEC	Word	Signal unit error counter, user initialized to 0. Incremented each time an SU is received that contains an error. These errors are: short frame, long frame, CRC error, and non-octet aligned error.
0x70	SS7STATE	Word	Internal state of SS7 controller, user initialized to 0.
0x74	JTSRTmp	Word	Temporary storage for time-stamp register value. Applies to Japanese SS7 only; otherwise should be cleared. Used by the CP to implement the 24-ms delay for signal unit error rate monitoring in Japanese SS7.
0x78	JTRDelay	Hword	FISU transmit delay (specified in units of 512us). Applies to Japanese SS7 only; otherwise should be cleared. According to the Japanese SS7 standard, the delay should be 24 ms and thus JTRDelay should be programmed to 24 ms/512 µs = 46.875 (approximately 47). Hence, the user should program JTRDelay to 0x2F and the RTSCR to generate a 1 µs time stamp period. Refer to Section 20.3.7, “RISC Time-Stamp Control Register (RTSCR),” in the *MPC8560 User’s Manual.*
0x7A	M	Hword	ITU threshold for AERM. If M_cnt reaches M, an AERM interrupt is generated. Note that M is normally programmed to 5.
0x7C	M_cnt	Hword	Up-counter for M. Should be cleared during initialization.

[1] The offset is relative to the dual-port RAM address + 64*CH_NUM. SS7 channel specific parameters require twice the amount of dual-port RAM required for HDLC or Transparent channel specific parameters. Therefore for SS7 even channel numbers (0, 2, 4, etc.) must be used and odd channel number must be left unused.

33.9.1 Extended Channel Mode Register (ECHAMR)

The extended channel mode register (ECHAMR) is a user-initialized register, shown in Figure 33-11 It includes both the interrupt mask bits and channel configuration bits.

The interrupt mask provides bits for enabling/disabling each event defined in the interrupt table entry. Other bits provide various channel configuration options.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	MODE0	—	OCT	SUERM	FISU	—	UN	TXB	—	AERM	NID	IDL	MRF	RXF	BSY	RXB
Reset	No reset value															
R/W	R/W															
Offset	0x18															

	16	17	18	19	20 — 25	26	27	28	29 — 31
Field	MODE1	POL	1	IDLM	—	TS	RQN		NOF
Reset	No reset value								
R/W	R/W								
Offset	0x1A								

Figure 33-11. Extended Channel Mode Register (ECHAMR)

ECHAMR fields are described in Table 33-10.

Table 33-10. ECHAMR Fields Description

Bits	Name	Description
0,16	MODE0 MODE1	00 Transparent mode 01 HDLC mode 10 Reserved 11 SS #7 mode (This is the required bit setting for an MCC to perform SS7.)
1, 5, 8	0	Reserved, should be cleared during initialization.
2-4 6-7 9-15	INTMSK	Interrupt mask bits, see Subsection 44.9.1, “Interrupt Table Entry”.
17	POL	Enable polling. POL enables the transmitter to poll the TxBDs. 0 Polling is disabled (The CPM does not access the external bus to check the R bit in the TxBD). 1 Polling is enabled. POL can be used to optimize the use of the external bus. Software should always set POL at the beginning of a transmit sequence of one or more frames. The CP clears POL when no more buffers are ready in the transmit queue, i.e. when it finds a BD with R = 0 (for example, at the end of a frame or at the end of a multi-frame transmission). To prevent a significant number of useless transactions on the external bus, software should always prepare the new BD, or multiple BDs, and set BD[R] before enabling polling.
18	1	Reserved, must be set.

Table 33-10. ECHAMR Fields Description (continued)

Bits	Name	Description
19	IDLM	Idle mode. 0 No idle patterns are transmitted between frames. After transmitting NOF+1 flags, the transmitter starts sending the data of the frame. If the transmission is between frames and the frame buffers are not ready, the transmitter sends flags until it can start transmitting the data. 1 At least one idle pattern is sent between adjacent frames. The NOF value shall be no smaller than the PAD setting, see TxBD. If NOF = 0, this is identical to flag sharing in SS7. Mode flags precede the actual data. When IDLM = 1, at least one idle pattern is sent between adjacent frames. If the transmission is between frames and the frame buffer is not ready, the transmitter sends idle characters. When data is ready, the NOF+1 flags are sent followed by the data frame. If IDLE mode is selected and NOF = 1, the following sequence is sent: .....init value, FF, FF, flag, flag, data, ... The init value before the idle will be ones.
20-25	—	Reserved, should be cleared during initialization.
10	TS	Receive time stamp. If this bit is set a 4 byte time stamp is written at the beginning of every data buffer that the BD points to.If this bit is set the data buffer must start from an address equal to 8*n-4 (n is any integer larger than 0).
11–12	RQN	Receive queue number. Specifies the receive interrupt queue number. 00 Queue number 0. 01 Queue number 1. 10 Queue number 2. 11 Queue number 3.
13–15	NOF	Number of flags. NOF defines the minimum number of flags before frames: 000 - at least 1 flag 001 - at least 2 flags .... 111 - at least 8 flags

Note that items in bold must be initialized by the user.

33.9.2 Signal Unit Error Monitor (SUERM)

The microcode maintains the signal unit error rate monitor as described in ITU-T Q.703 paragraph 10, and ANSI T1.111-1996 paragraph 10.

The microcode uses SUERM, N, N_cnt, D, D_cnt and T parameters for the leaky-bucket implementation of the SU error monitor.

- After every N octets received while in octet counting mode, SUERM is incremented and an interrupt request can be generated (SUERM) depending on the interrupt mask.
- After D error-free frames have been received, SUERM is decremented. SUERM will not be decremented below zero.
- If SUERM reaches T, the SUERM is cleared and an interrupt is generated.

33.9.3 SUERM in Japanese SS7

The Japanese SS7 uses a time interval to monitor errors. If an error is present, it checks every 24 ms.

- An error flag is set that indicates whether current frame is errored or not.
- For every JTRDelay an error flag is checked.
- If there is no error, decrement the counter SUERM by 1 (not below zero).
- If there is an error, increment the counter SUERM by D.
- If SUERM reaches T, the counter SUERM is cleared and a “signal unit error rate monitor” interrupt is generated.

Table 33-11. Parameter Values for SUERM in Japanese SS7

Parameter	Definition	Value
T	Threshold	285
D	Upcount	16
JTRDelay	Length of interval (24ms)	0x2F

33.9.4 SS7 Configuration Register (SS7_OPT)

The SS7 configuration register, shown on Figure 33-12 contains additional SS7 parameters.

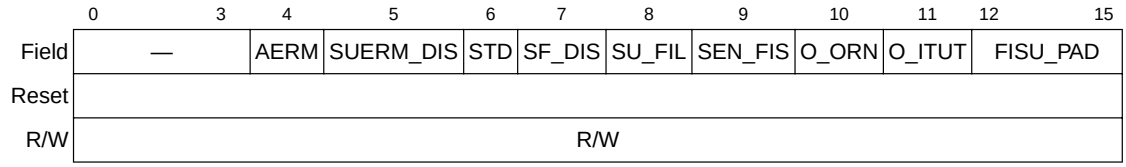


Figure 33-12. SS7 Configuration Register (SS7_OPT)

Table 33-12 describes SS7 configuration register fields.

Table 33-12. SS7 Configuration Register Fields Description.

Bits	Name	Description
0-3	—	Reserved, should be cleared during initialization.
4	AERM	Alignment error rate monitor enable. See Section 33.9.4.1, “AERM Implementation.” 0 Do not enable AERM. 1 Enable AERM.
5	SUERM_ DIS	Disable the SU error rate monitor. See Section 33.9.4.3, “Disabling SUERM.” 0 Enable SUERM. 1 Disables both SUERM and AERM.
6	STD	Standard compliance 0 ITU-T/ANSI compliant 1 Japanese SS7 compliant

Table 33-12. SS7 Configuration Register Fields Description.

Bits	Name	Description
7	SF_DIS	Discard short frames (less than 5 octets) 0 Do not discard short frames. 1 Discard short frames.
8	SU_FIL	SU Filtering 0 Disable SU filtering. 1 Enable SU filtering.
9	SEN_FIS	Send FISU if first BD of frame is not ready. 0 Flags are sent if the current BD, which is the first BD of the frame, does not have its ready bit set. 1 FISUs are automatically sent if the current BD, which is the first BD of the frame, does not have its ready bit set.
10	O_ORN	Enter octet counting mode (OCM) on overrun. Should be cleared if using the Japanese standard. 0 Disable entering OCM if there are no receive BDs available. 1 Enter OCM if there are no receive BDs available. Note that when STD = 1, O_ORN = 1, and no receive buffers are ready, a ny received signal unit is treated as an erred signal unit.
11	O_ITUT	Enter octet counting mode (OCM) on ITU-T conditions (after an abort sequence or when an SU is too long). Should be cleared if using the Japanese standard. 0 Disable entering OCM on ITU-T conditions. 1 Enable entering OCM on ITU-T conditions.
12-15	FISU_PAD	Padding of the automatically transmitted FISUs. If the SEN_FISU bit is set, the CP will use the value of FISU_PAD as a number of pad character. Please refer to PAD parameter in 44.10.2, “SS7 Transmit Buffer Descriptor (TxBD).

33.9.4.1 AERM Implementation

The SS7 microcode implements the ITU Q.703 alignment error rate monitor (AERM). The microcode uses the T, SUERM, M and M_cnt parameters. The M_cnt parameter is incremented for every T errored frames. If M_cnt reaches M, an AERM interrupt is generated to layer 3.

Note that in AERM mode no SUERM interrupt is generated. Also, the algorithm associated with D and D_cnt is disabled as per the ITU specification.

33.9.4.2 Japanese SS7

To meet the Japanese AERM requirements the user must change the parameters T and D. Note that the interrupt generated is not AERM but SUERM.

During proving, do the following: set parameters T (Threshold) and D (up counter) to ‘1.’ SUERM should initially be cleared to ‘0,’ so that an interrupt will be generated at the first error.

After proving period, set the parameters (T and D) to values according to the Japanese SUERM. See section Table 33-11.

To disable AERM and enter SUERM, do the following:

1. Set SUERM_DIS bit in SS7_OPT.
2. Set parameters (T, D & SUERM) for Japanese SUERM.
3. Clear SUERM_DIS bit in SS7_OPT.

33.9.4.3 Disabling SUERM

When SS7_OPT[SUERM_DIS] is set, the N_cnt and D_cnt parameters are not decremented by the microcode and no SUERM interrupt is generated. This allows these parameters to be updated, for example, at the end of the proving period in alignment error monitoring.

Note: If the SS7 controller is in the octet counting mode (OCM) when SUERM_DIS is set, then if no idles (only flags/data) are received while SUERM_DIS is set, then after the host updates N_cnt and D_cnt and clears SUERM_DIS the receiver will start in OCM. However if SUERM_DIS is set while in OCM and idles are then received then the OCM state is left.

33.9.4.4 SU Filtering

In order to reduce the overhead to the user software, a filtering algorithm has been adopted to allow superfluous frames to be discarded. This algorithm compares the first 3–5 bytes (depending on the type) of the current FISU or LSSU to the last SU received and discards the current SU if it has already been received twice.

33.9.4.4.1 Comparison Mask

A user programmable 5-byte mask exists in the parameter RAM map. When an SU is received, the controller checks the contents of the LI field. If LI is between 0 and 2, the SU (except for the CRC portion) will be masked according to the user programmable mask and will then be compared to the last SU received. The state machine for the matching algorithm is in Subsection 33.9.4.4.2, “Comparison State Machine”.

The Mask1 and Mask2 channel-specific parameters construct the 5-byte user mask. The exact format and byte ordering are shown on Figure 33-13 and Figure 33-14.

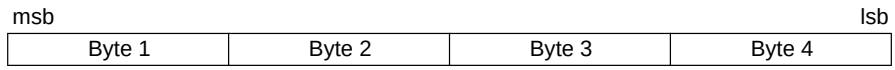


Figure 33-13. Mask1 Format

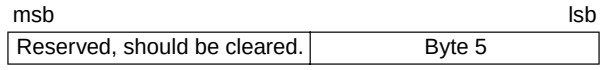


Figure 33-14. Mask2 Format

33.9.4.4.2 Comparison State Machine

The state machine shown in Figure 33-15 exists for filtering.

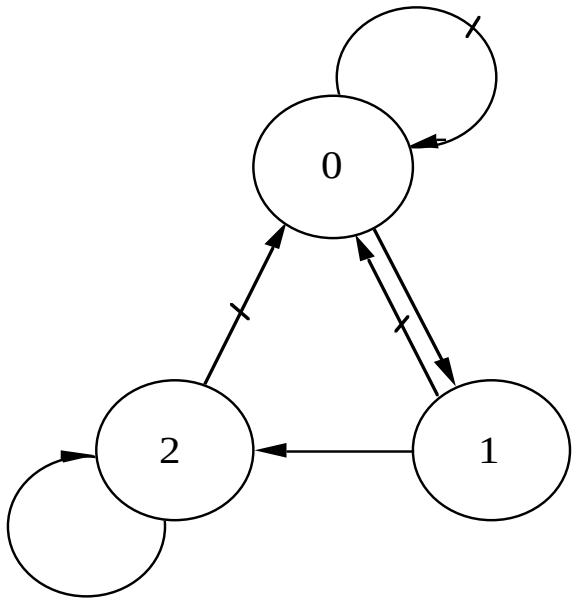


Figure 33-15. Filtering State Machine

- State 0—The first 3-5 bytes (depending on the contents of the LI field) are masked and then compared with the first 3-5 bytes of the last SU. If there is a match, go to State 1, else remain in State 0. The current SU will be received into a buffer descriptor.
- State 1—The first 3-5 bytes (depending on the contents of the LI field) are masked and then compared with the first 3-5 bytes of the last SU. If there is a match, go to State 2, else go to State 0. The current SU will be received into a buffer descriptor.
- State 2—The first 3-5 bytes (depending on the contents of the LI field) are masked and then compared with the first 3-5 bytes of the last SU. If there is a match, the current SU will be discarded (unless there is an error), the channel will remain in state 2 and SU error monitor will be adjusted accordingly. If the frames do not match, the current SU will be received into a buffer descriptor and the channel will return to State 0.

33.9.4.4.3 Filtering Limitations

Because the algorithm is purely checking identical SUs, two FISUs will be received after each MSU rather than merely one, even though they have the same sequence numbers. Reception of an MSU resets the filtering algorithm. Also, reception of a short frame resets the filtering algorithm when SS7_OPT[SF_DIS] = 0; however, when

SS7_OPT[SF_DIS] = 1 (short frames are discarded), the filtering algorithm remains unchanged.

33.9.4.4.4 Resetting the SU Filtering Mechanism

This command resets the filtering algorithm to ensure that the next SU will be received, even if it would normally have been filtered. This command could be issued periodically so that the 603e core can check to make sure that the link is really up and not simply receiving flags.

To issue this MCC command, refer to Section 20.4, “Command Set,” in the *MPC8560 User’s Manual*. Use opcode 1110 (0xE).

33.9.4.5 Octet Counting Mode

When entering the octet counting mode (OCM), the CP will load the user defined N register to its internal octet counter. While in the octet counting mode the CP will decrement its internal counter for every unstuffed octet received. When the internal counter is decremented to zero, the CP increment the SUERM register and reload the N register into the internal count register. In addition an interrupt (OCT) might be generated depending on the interrupt mask. The SS7 controller will enter octet counting mode under the following circumstances:

- An ABORT character is received at any time and SS7_OPT[O_ITUT] is set.
- The SU currently being received has exceeded the length programmed in the MFLR register and SS7_OPT[O_ITUT] is set.
- The receiver overruns and SS7_OPT[O_ORN] is set. Note that when no receive buffers are available, only octets are counted; that is, D_cnt is not decremented after receiving the frame.

The SS7 controller will leave octet counting mode when a valid signal unit is detected (with a valid CRC and a length less than MFLR and greater than 4).

NOTE

Octet counting mode applies only to the ITU-T and ANSI standards. The SS7 microcode will not work if both the Japanese standard and OCM features are selected.

33.10 MCC Configuration Registers (MCCF*x*)

The MCC configuration register (MCCF), shown in Figure 33-16, defines the mapping of the MCC channels to the TDM channels. MCC1 can be connected to SI1 and MCC2 can be connected to SI2. For each MCC*x*-SI*x* pair, each of the four 32 channels subgroups can be connected to one of the four TDM highways (TDMA, TDMB, TDMC, and TDMD).

	0	1	2	3	4	5	6	7
Field	Group 1		Group 2		Group 3		Group 4	
Reset	0000_0000							
R/W	R/W							
Addr	0x9_1B38 (MCCF1), 0x9_1B58 (MCCF2)							

Figure 33-16. SI MCC Configuration Register (MCCF)

Table 33-13 describes MCCF fields.

Table 33-13. MCCF Field Descriptions

Bits	Name	Description
0–1, 2–3, 4–5, 6–7	GROUP *x*	Group x of channels is used by TDM y as shown in Table 33-14. 00 Group x is used by TDM A. 01 Group x is used by TDM B. 10 Group x is used by TDM C. 11 Group x is used by TDM D.

Table 33-14 describes group assignments.

Table 33-14. Group Channel Assignments

Group	Channels
Group1 in MCCF1	0–31
Group2 in MCCF1	32–63
Group3 in MCCF1	64–95
Group4 in MCCF1	96–127
Group1 in MCCF2	128–159
Group2 in MCCF2	160–191
Group3 in MCCF2	192–223
Group4 in MCCF2	224–255

Note that the TDM group channel assignments made in MCCF must be coherent with the SI register programming and SI RAM programming; see Section 22.5, “Serial Interface Registers,” and Section 22.4.3, “Programming SIx RAM Entries.” The user must also program MCCF before enabling the TDM channel in the SIGMR; see Section 22.5.1, “SI Global Mode Registers (SIxGMR).”

33.11 MCC Commands

The user starts channels by writing to the TSTATE/RSTATE registers as described in Section 33.7.4, “Internal Receiver State (RSTATE),” and Section 33.7.1, “Internal Transmitter State (TSTATE).”

The following commands, used to stop and initialize channels, are issued to the MCC by writing to CPCR as described in Section 20.4.1, “CP Command Register (CPCR).” Table 33-15 describes transmit commands.

Table 33-15. Transmit Commands

Command	Description
STOP TRANSMIT	Disables the transmission on the selected channel and clears CHAMR[POL]. When this command is issued in the middle of a frame, the CP sends an ABORT indication and then idles/flags on the selected channel. If this command is issued between frames, the CP sends only idles or flags (depending on CHAMR[IDLM]). TBPTR points for the buffer that the CP was using when the STOP TRANSMIT command was issued.
INIT TX PARAMETERS[1]	Initializes transmit parameters in this MCC parameter RAM to their reset state. This command should be issued at initialization while the transmitter is disabled. Note that the MCC initialize commands initialize only the 32 consecutive channels starting with the channel number specified in CPCR[MCN]. To initialize more than 32 channels, reissue the command with the appropriate channel numbers. Note also the INIT TX AND RX PARAMETERS command can be used to reset both the receive and transmit parameters.

1 The init parameters style commands are also used to reset the MCC channel FIFOs and these commands need to be issued to cover any channel number used, whether used normally or as part of a superchannel

Table 33-16 describes receive commands.

Table 33-16. Receive Commands

Command	Description
STOP RECEIVE	Forces the receiver of the selected channel to terminate reception. After this command is executed, the CP does not change the receive parameters in the dual-port RAM. The user must initialize the channel receive parameters in order to restart reception.
INIT RX PARAMETERS	Initializes transmit parameters in this MCC parameter RAM to their reset state. This command should be issued at initialization and when the transmitter is disabled. Note that the MCC initialize commands initialize only the 32 consecutive channels starting with the channel number specified in CPCR[MCN]. To initialize more than 32 channels, reissue the command with the appropriate channel numbers. Note also the INIT TX AND RX PARAMETERS command can be used to reset both the receive and transmit parameters.

33.12 MCC Exceptions

MCC interrupt handling involves two main data structures, the MCC event register (described in Section 33.12.1, “MCC Event Register (MCCE)/Mask Register (MCCM)) and the interrupt circular tables, shown in Figure 33-17.

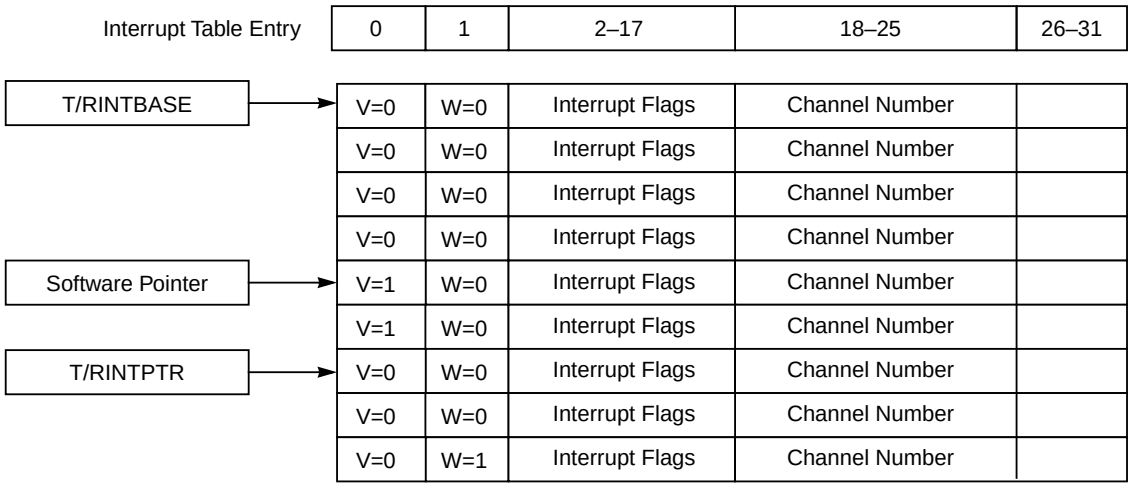


Figure 33-17. Interrupt Circular Table

There is one table for transmitter interrupts and from one to four tables for receiver interrupts. Each channel is programmed to report receiver interrupts in one of the receiver tables. This way receiver interrupts can be sorted, for example, by priority. Each interrupt circular table must be least two entries long.

T/RINTBASE and T/RINTPTR, which are user-initialized global MCC parameters (See Section 33.4, “Global MCC Parameters”), point to the starting location of the table (in external memory) and the current empty position (initialized at the top of the table) available to the CP. All the entries in the table must be user-initialized with 0x00000000, except for the last one which must be initialized with 0x40000000 (W = 1, thus defining the end of the table). When an MCC channel generates an interrupt request, the CP writes a new entry to the table (with V = 1) and increments T/RINTPTR (if W = 1 for the current entry, T/RINTPTR is loaded with T/RINTBASE).

An interrupt is issued to the core whenever an entry is added to an interrupt circular table, except for the RXF events (received complete HDLC frame), in which case an interrupt is issued after a total of GRFTHR entries were added to one or more of the receive interrupt circular tables. See Table 33-1 for the description of the GRFTHR.

In addition to the channel’s number, this entry contains a description of the exception (see Section 33.12.1.1, “Interrupt Table Entry”).

After an MCC interrupt, the user reads MCCE. MCCE[GINT] can be used to indicate that at least one new entry was added to one of the tables. After clearing GINT, the user starts processing the table(s) which contain pending events, as indicated by the bits MCCE[RINT*x*] and MCCE[TINT]. The user then clears this entry’s valid bit (V) (see Section 33.12.1.1, “Interrupt Table Entry”). The user follows this procedure until it reaches an entry with V = 0.

33.12.1MCC Event Register (MCCE)/Mask Register (MCCM)

The MCC event register (MCCE) is used to report events and generate interrupt requests. For each of its flags, a programmable mask/enable bit in MCCM determines whether an interrupt request is generated. The MCC mask register (MCCM) is used to enable/disable interrupt requests. For each flag in the MCCE there is a programmable mask/enable bit in MCCM which determines whether an interrupt request is generated. Setting an MCCM bit enables and clearing an MCCM bit disables the corresponding interrupt.

MCCE bits are cleared by writing ones to them; writing zeros has no effect.

Figure 33-18 shows MCCE and MCCM bits.

	0	1	2	3	4	5	6	7	8 — 11	12	13	14	15
Field	QOV0	RINT0	QOV1	RINT1	QOV2	RINT2	QOV3	RINT3	—	TQOV	TINT	GUN	GOV
Reset	0000_0000_0000_0000												
R/W	R/W												
Addr	0x9_1B30 (MCCE1), 0x9_1B50 (MCCE2)/0x9_1B34 (MCCM1), 0x9_1B54 (MCCM2)												

Figure 33-18. MCC Event Register (MCCE)/Mask Register (MCCM)

Table 33-17 describes MCCE fields.

Table 33-17. MCCE/MCCM Register Field Descriptions

Bits	Name	Description
0	QOV0	QOVx—Receive interrupt queue overflow. IQOV is set (and an interrupt request generated) by the CP whenever an overflow occurs in the transmit circular interrupt table. This occurs if the CP tries to update an interrupt entry that was not handled by the user (such an entry is identified by V = 1). RINTx—Receive interrupt. When RINT = 1, the MCC generated at least one new entry in the receive interrupt circular table. After clearing it, the user reads the next entry from the receive interrupt circular table and starts processing a specific channel's exception. The user returns from the interrupt handler when it reaches a table entry with V = 0.
1	RINT0	
2	QOV1	
3	RINT1	
4	QOV2	
5	RINT2	
6	QOV3	
7	RINT3	
8–11	—	Reserved, should be cleared.
12	TQOV	Transmit interrupt queue overflow. TQOV is set (and interrupt request generated) by the CP whenever an overflow occurs in the transmit circular interrupt table. This condition occurs if the CP attempts to write a new interrupt entry into an entry that was not handled by the user. Such an entry is identified by V = 1.
13	TINT	Transmit interrupt. When TINT = 1, at least one new entry in the transmit interrupt circular table was generated by MCC. After clearing it, the user reads the next entry from the transmit interrupt circular table and starts processing a specific channel's exception. The user returns from the interrupt handler when it reaches a table entry with V = 0.

Table 33-17. MCCE/MCCM Register Field Descriptions (continued)

Bits	Name	Description
14	GUN	Global transmitter underrun. When set, this flag indicates that an underrun occurred in the MCC's transmitter FIFO buffer. This error is fatal, since it is unknown which channels were affected. Following the assertion of GUN in the MCCE the MCC stops transmitting data in all channels. The TDM Tx line becomes idle. After this error, the SI should be stopped, the CP reset, and then the MCC channels reinitialized. If not masked in MCCM, an interrupt request is generated when GUN is set. The user must clear GUN. Note: Although the aggregate serial rate is within capacity, an MCC transmitter underrun condition may occur when the SI frame length is shorter than the gap between sync signals. This situation can be avoided by adding dummy SI RAM entries with unsupported bits (MCC=0 and CSEL = 0000) at the end of shorter SI frames such that the frame length matches the gap between syncs.
15	GOV	Global receiver overrun. When GOV = 1, an overrun occurred in the MCC receiver FIFO buffer. This error is fatal, since it is unknown which channels were affected. When GOV = 1, the MCC stops receiving data in all channels. No more data is transferred to memory. The MCC receivers must be re-initialized after this error. If enabled in MCCM, an interrupt request is generated when GOV is set. The user must clear GOV bit.

33.12.1.1 Interrupt Table Entry

Each interrupt table entry, shown in Figure 33-19, contains information about channel-specific events. The transmit circular table shows only events caused by transmission; the receive circular tables shows only events caused by reception.

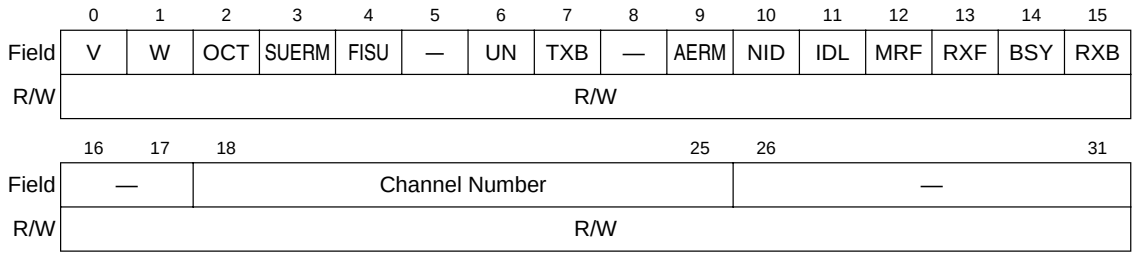


Figure 33-19. Interrupt Circular Table Entry

Table 33-18 describes interrupt circular table fields.

Table 33-18. Interrupt Circular Table Entry Field Descriptions

Bits	Name	Description
0	V	Valid bit. V = 1 indicates that this entry contains valid interrupt information. Upon generating a new entry, the CP sets V = 1. The user clears V immediately after it reads the interrupt flags of the entry (before processing the interrupt). The V bits in the table are user-initialized. During initialization, the user must clear those bits in all table entries.
1	W	Wrap bit. W = 1 indicates the last interrupt circular table entry. The next event's entry is written/read (by CP/user) from the address contained in INTBASE (see Table 33-1). During initialization, the user must clear all W bits in the table except for the last one which must be set.
2	OCT	For SS7 only; otherwise reserved. N octects received. If the channel is in octect counting, this bit is set when N octects have been received.

Table 33-18. Interrupt Circular Table Entry Field Descriptions (continued)

Bits	Name	Description
3	SUERM	For SS7 only; otherwise reserved. SU error monitor threshold reached. The SU error monitor has reached the program threshold T.
4	FISU	For SS7 only; otherwise reserved. FISU transmission started. The CP has started automatic FISU transmission if the first BD of frame does not have its ready bit set and the SEN_FISU bit is enabled in SS7_OPT register. Please refer to SEN_FISU bit, SS7 Configuration Register (SS7_OPT).
5	—	Reserved, should be cleared.
6	UN	Tx no data. The CP sets this flag if there is no data available to be sent to the transmitter. The transmitter sends an ABORT indication and then sends idles.
7	TXB	Tx buffer. A buffer has been completely transmitted. TXB is set (and an interrupt request is generated) as soon as the programmed number of PAD characters (or the closing flag, for PAD = 0) is written to MCC transmit FIFO. This controls when the TXB interrupt is given in relation to the closing flag sent out at TXD. Section 33.13.2, “Transmit Buffer Descriptor (TxBD)” describes how PAD characters are used.
8	—	Reserved, should be cleared.
9	AERM	For SS7 only; otherwise reserved. The alignment error rate monitor has reached the programmed threshold M."
10	NID	Set whenever a pattern that is not an idle pattern is identified.
11	IDL	Idle. Set when the channel’s receiver identifies the first occurrence of HDLC idle (0xFFFE) after any non-idle pattern.
12	MRF	Maximum receive frame length violation. This interrupt occurs in HDLC mode when more bytes are received than the value specified in MFLR. This interrupt is generated as soon as the MFLR value is exceeded; the remainder of the frame is discarded
13	RXF	Rx frame. A complete HDLC frame has been received.
14	BSY	Busy. A frame was received but was discarded due to lack of buffers.
15	RXB	Rx buffer. A buffer has been received on this channel that was not the last buffer in frame. This interrupt is also given for different error types that can happen during reception. Error conditions are reported in the RxBD.
16–17	—	Reserved, should be cleared.
18–25	CN	Channel number. Identifies the requests channel index (0–255).
26–31	—	Reserved, should be cleared.

33.13 MCC Buffer Descriptors

Each MCC channel requires two BD tables (one for transmit and one for receive). Each BD contains key information about the buffer it defines. The BDs are accessed by the MCC as needed; BDs can be added dynamically to the BDs chain. The RxBDs chain must include at least two BDs; the TxBD chain must include at least one BDs.

The MCC BDs are located in the external memory.

The SS7 RxBDs have an additional status flag for short frames (RxBD[SF]), and the SS7 TxBDs have a retransmit control bit (TxBD[RT]). The following sections describe the BD structures for SS7 operation.

33.13.1Receive Buffer Descriptor (RxBD)

Figure 33-20 shows the RxBD.

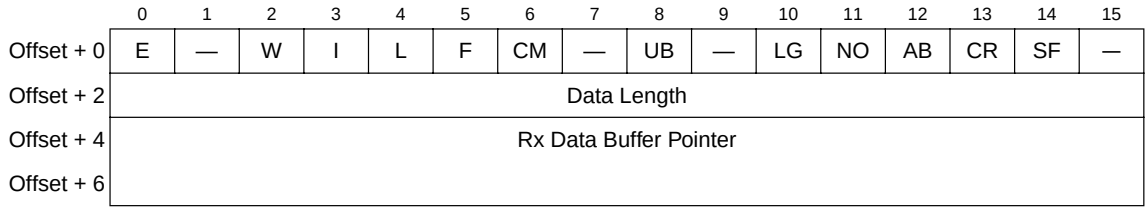


Figure 33-20. MCC Receive Buffer Descriptor (RxBD)

RxBD fields are described in Table 33-19.

Table 33-19. RxBD Field Descriptions

Bits	Name	Description
0	E	Empty 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 user is free to examine or write to any fields of this RxBD. The CP does not use this BD again while the empty bit remains zero. 1 The data buffer associated with this BD is empty, or reception is in progress. This RxBD and its associated receive buffer are in use by the CP. When E = 1, the user should not write any fields of this RxBD.
1	—	Reserved, should be cleared.
2	W	Wrap (final BD in table) 0 This is not the last BD in the RxBD table. 1 This is the last BD in the RxBD table. After this buffer has been used, the CP receives incoming data into the first BD in the table (the BD pointed to by RBASE). The number of RxBDs in this table is programmable and is determined by the wrap bit.
3	I	Interrupt 0 The RXB bit is not set after this buffer has been used, but RXF operation remains unaffected. 1 The RXB or RXF bit in the HDLC interrupt circular table entry is set when this buffer has been used by the HDLC controller. These two bits may cause interrupts (if enabled).
4	L	Last in frame (only for HDLC mode of operation). The HDLC controller sets L = 1, when this buffer is the last in a frame. This implies the reception either of a closing flag or of an error, in which case one or more of the CD, OV, AB, and LG bits are set. The HDLC controller writes the number of frame octets to the data length field. 0 This buffer is not the last in a frame. 1 This buffer is the last in a frame.
5	F	First in frame. The HDLC controller sets F = 1 for the first buffer in a frame. In transparent mode, F indicates that there was a synchronization before receiving data in this BD. 0 This is not the first buffer in a frame. 1 This is the first buffer in a frame.

Table 33-19. RxBD Field Descriptions (continued)

Bits	Name	Description
6	CM	Continuous mode 0 Normal operation (The empty bit (bit 0) is cleared by the CP after this BD is closed). 1 The empty bit (bit 0) is not cleared by the CP after this BD is closed, allowing the associated data buffer to be overwritten automatically when the CP next accesses this BD. However, if an error occurs during reception, the empty bit is cleared regardless of the CM bit setting.
7	—	Reserved, should be cleared.
8	UB	User bit. UB is a user-defined bit that the CPM never sets nor clears. The user determines how this bit is used.
9	—	Reserved, should be cleared.
10	LG	Rx frame length violation (HDLC mode only). Indicates that a frame length greater than the maximum value was received in this channel. Only the maximum-allowed number of bytes, MFLR rounded to the nearest higher word alignment, are written to the data buffer. This event is recognized as soon as the MFLR value is exceeded when data is word-aligned. When data is not word-aligned, this interrupt occurs when the SDMA writes 64 bits to memory. The worst-case latency from MFLR violation until detected is 7 bytes timing for this channel. When MFLR violation is detected, the receiver is still receiving even though the data is discarded. The buffer is closed upon detecting a flag, and this is considered to be the closing flag for this buffer. At this point, LG is set (1) and an interrupt may be generated. The length field for this buffer is everything between the opening flag and this last identifying flag.
11	NO	Rx nonoctet-aligned frame. A frame of bits not divisible exactly by eight was received. NO = 1 for any type of nonalignment regardless of frame length. The shortest frame that can be detected is of type FLAG-BIT-FLAG, which causes the buffer to be closed with NO error indicated. The following shows how the nonoctet alignment is reported and where data can be found. msb lsb xxx ..... xx | 1 | 000..... 0 Valid Data | Invalid Data To accommodate the extra word of data that may be written at the end of the frame, it is recommended to reserve MFLR + 8 bytes for each buffer data.
12	AB	Rx abort sequence. A minimum of seven consecutive 1s was received during frame reception. Abort is not detected between frames. The sequence Closing-Flag, data, CRC, AB, data, opening-flag... does not cause an abort error. If the abort is long enough to be an idle, an idle line interrupt may be generated. An abort within the frame is not reported by a unique interrupt but rather with a RXF interrupt and the user has to examine the BD.
13	CR	Rx CRC error. This frame contains a CRC error. The received CRC bytes are always written to the receive buffer.
14	SF	For SS7 only; otherwise reserved. Short frame indication. Set if the received frame is less than 5 octects.
15	—	Reserved, should be cleared.

The data length and buffer pointer are described as follows:

- Data length. Data length is the number of octets written by the CP into this BD's data buffer. It is written by the CP when the BD is closed. When this is the last BD in the frame (L = 1), the data length contains the total number of frame octets (including

two or four bytes for CRC). Note that memory allocated for buffers should be not smaller than the contents of the maximum receive buffer length register (MRBLR). The data length does not include the time stamp.

- Rx buffer pointer. The receive buffer pointer points to the first location of the associated data buffer. This value must be equal to $8 \times n$ if CHAMR[TS] = 0 and equal to $8 \times n$ - 4 if CHAMR[TS] = 1 (where *n* is any integer larger than 0). For SS7, this value must be equal to 8*n if ECHAMR[TS]=0 and equal to 8*n-4 if ECHAMR[TS]=1 (n = any integer larger than 0).

33.13.2 Transmit Buffer Descriptor (TxBD)

Figure 33-21 shows the TxBD.

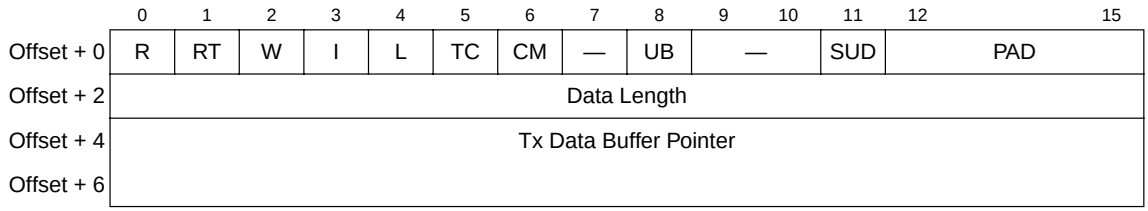


Figure 33-21. MCC Transmit Buffer Descriptor (TxBD)

Table 33-20 describes TxBD fields.

Table 33-20. TxBD Field Descriptions

Bits	Name	Description
0	R	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The CP clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer is ready to be transmitted. The transmission may have begun, but it has not completed. The user cannot modify this BD once this bit is set.
1	RT	For SS7 only; otherwise reserved. Retransmit 0 Normal operation 1 The CP will repeat transmission of this BD until the RT bit is cleared. After the RT bit is cleared the CP advances to the next BD in the table. This feature is useful for automatic LSSU retransmission.
2	W	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer is used, the CP receives incoming data into the first BD in the table (the BD pointed to by TBASE). The number of TxBDs in this table is programmable and is determined the wrap bit.
3	I	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 TXB in the circular interrupt table entry is set when this buffer has been serviced by the MCC. This bit can cause an interrupt (if enabled).
4	L	Last 0 This is not the last buffer in the frame. 1 This is the last buffer in the current frame.

Table 33-20. TxBD Field Descriptions (continued)

Bits	Name	Description
5	TC	Tx CRC. Valid only when L = 1. Otherwise it is ignored. 0 Transmit the closing flag after the last data byte. This setting can be used for testing purposes to send an erroneous CRC after the data. 1 Transmit the CRC sequence after the last data byte.
6	CM	Continuous mode 0 Normal operation. 1 The CP does not clear the ready bit after this BD is closed, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of the CM bit setting.
7	—	Reserved, should be cleared.
8	UB	User bit. UB is a user-defined bit that the CPM never sets nor clears. The user determines how this bit is used.
9–10	—	Reserved, should be cleared.
11	SUD	For SS7 only; otherwise reserved. Signal unit delay 0 This buffer does not have a transmission delay. 1 A time delay of JTTDelay x 512 µs passes before this buffer is transmitted. Can be used for LSSU transmission according to the JT Q.703 Standard which defines a 24 ms delay between back-to-back LSSUs. This bit is only valid when SS7_OPT[STD] is set.
12–15	PAD	Pad characters. These four bits indicate the number of PAD characters (0x7E or 0xFF depending on the IDLM mode selected in the CHAMR register) that the transmitter sends after the closing flag. The transmitter issues a TXB interrupt only after sending the programmed number of pads to the Tx FIFO buffer. The user can use the PAD value to guarantee that the TXB interrupt occurs after the closing flag has been sent out on the TXD line. PAD = 0, means that the TXB interrupt is issued immediately after the closing flag is sent to the Tx FIFO buffer. The number of PAD characters depends on the FIFO size assigned to the channel in the MCC hardware. If the channel is not part of a super channel then the MCC hardware assigns to this channel a fifo of 4 bytes. So in this case a pad of 4 bytes ensure that the TXB interrupt is not given before the closing flag has been transmitted over the TXD line. For a super channel, the FIFO length equals the number of time slots assigned to the super channel multiplied by two.

The data length and buffer pointer are described below:

- Data length. The data length is the number of bytes the MCC should transmit from this BD’s data buffer. It is never modified by the CP. The value of this field should be greater than zero.
- Tx buffer pointer. The transmit buffer pointer, which contains the address of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory. This value is never modified by the CP. For SS7, the transmit buffer pointer, which contains the address of the associated data buffer, may be even or odd provided that SS7_OPT[SEN_FIS] = 0. If the automatic FISU option is required then the buffer pointer must be 4-byte aligned."

33.14 MCC Initialization and Start/Stop Sequence

The MCC must be initialized and started/stopped in relation with the corresponding TDMs. The following two sections present the initialization and start/stop sequences which must be followed for single and super channels.

33.14.1 Single-Channel Initialization

The following sequence must be followed to initialize and start a single channel (after reset or after a fatal error):

1. Program the SI. The entries the MCC channels uses must point to the null channel (set in the SI RAM entry MCC = 0, CSEL = 0 and the correct size - 1 byte); entries used by other controllers (not MCC) can be activated at this time.
2. Initialize the MCC parameters (in DPR and external memory).
3. Enable the MCC channel as described in Section 33.7.1, “Internal Transmitter State (TSTATE),” and Section 33.7.4, “Internal Receiver State (RSTATE).”
4. Reprogram the SI RAM to point to the enabled channel(s).

The following sequence must be followed to stop a single channel in order to change the SI without using the shadow SI:

1. Issue a STOP command for the respective channel as described in Section 33.11, “MCC Commands.”
2. Change the SI.
3. Enable the MCC channel(s) as described in Section 33.7.1, “Internal Transmitter State (TSTATE),” and Section 33.7.4, “Internal Receiver State (RSTATE).”

It is possible to change the SI using the SI shadow while the channel is active. Both the primary and the shadow configuration of the SI RAM must observe the configuration defined in MCCF (see Section 33.10, “MCC Configuration Registers (MCCFx)”). The MCCF cannot be changed while there are active channels.

The following sequence must be followed to stop a single channel in order to change the MCC parameters of the respective channel:

1. Issue a STOP command for the respective channel as described in Section 33.11, “MCC Commands,” or change the associated SI RAM entry to point to a channel which is not active and wait for two frame periods in order to clear the internal FIFOs.
2. Change the channel parameters.
3. Enable the MCC channel(s) as described in Section 33.7.1, “Internal Transmitter State (TSTATE),” and Section 33.7.4, “Internal Receiver State (RSTATE),” or change the associated SI RAM entry to point to the respective channel.

33.14.2 Super Channel Initialization

The following steps initialize and start a super channel (after reset or after a fatal error):

1. Program the SI as required for a super channel but do not enable the TDM.
2. Issue a STOP command as described in Section 33.11, “MCC Commands.”
3. Enable the TDM.
4. Initialize the MCC parameters (in DPR and external memory).
5. Enable the MCC channel(s) as described in Section 33.7.1, “Internal Transmitter State (TSTATE),” and Section 33.7.4, “Internal Receiver State (RSTATE).”

The following sequence must be followed to stop a super channel in order to change the SI:

1. Issue a STOP command for the respective channel as described in Section 33.11, “MCC Commands.”
2. Disable the TDM.
3. Change the SI.
4. Enable the TDM.
5. If necessary, change the MCC parameters (in DPR and external memory).
6. Enable the MCC channel(s) as described in Section 33.7.1, “Internal Transmitter State (TSTATE),” and Section 33.7.4, “Internal Receiver State (RSTATE).”

Under the following restrictions, the SI can be changed using the SI shadow while the channel is active:

- Both the primary and the shadow configuration of the SI RAM must observe the configuration of the super channel. Note that the super-channel table and MCCF register cannot be changed dynamically.
- It is not possible to add dynamically to a super channel a time slot previously used by a single-channel and had a width different from 8 bits.
- A time slot that was previously used by a single channel and had a width different from 8 bits cannot be added dynamically to a super channel.

33.15 MCC Latency and Performance

The MCC transfers data to/from the memory 8 bytes at a time. Considering this and the internal receiver FIFO (of 2 bytes/channel), the receiver latency (time since data on a channel is serialized until the respective data is written to memory) can be from 8–10 frame periods.

If no super channels are active, the MCC can handle aggregate data rates of up to 16 Mbps on each of the four channel subgroups. If super channels are used this performance is limited to 8 Mbps.

If multiple synchronized channels are used (as an example 8 T1 with common clock/sync) it is recommended to start the channels out of phase in order to load uniformly the bus. This avoids bus activity peaks when all the channels have to transfer data to/from the memory simultaneously.

Chapter 34
Fast Communications Controllers (FCCs)

The MPC8560's fast communications controllers (FCCs) are serial communications controllers (SCCs) optimized for synchronous high-rate protocols. FCC key features include the following:

- Supports HDLC/SDLC and totally transparent protocols
- FCC clocks can be derived from a baud-rate generator or an external signal.
- Supports $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ modem control signals
- Use of bursts to improve bus usage
- Multibuffer data structure for receive and transmit, external buffer descriptors (BDs) anywhere in system memory
- 192-byte FIFO buffers
- Full-duplex operation
- Fully transparent option for one half of an FCC (receiver/transmitter) while HDLC/SDLC protocol executes on the other half (transmitter/receiver)
- Echo and local loopback modes for testing
- Assuming a 100-MHz CPM clock, the FCCs support the following:
  - — Full 10/100-Mbps Ethernet/IEEE 802.3x through an MII or RMII
  - — Full 155-Mbps ATM segmentation and reassembly (SAR) through UTOPIA (on FCC1 and FCC2 only)
  - — 45-Mbps (DS-3/E3 rates) HDLC and/or transparent data rates supported on each FCC

FCCs differ from SCCs as follows:

- No DPLL support.
- No BISYNC, UART, or AppleTalk/LocalTalk support.
- No HDLC bus.

34.1 Overview

MPC8560 FCCs can be configured independently to implement different protocols. Together, they can be used to implement bridging functions, routers, and gateways, and to interface with a wide variety of standard WANs, LANs, and proprietary networks. FCCs have many physical interface options such as interfacing to TDM buses, ISDN buses, standard modem interfaces, fast Ethernet interface (MII), and ATM interfaces (UTOPIA); see Chapter 22, “Serial Interface with Time-Slot Assigner,” Chapter 38, “Fast Ethernet Controller,” and Chapter 35, “ATM Controller.” The FCCs are independent from the physical interface, but FCC logic formats and manipulates data from the physical interface. That is why the interfaces are described separately.

The FCC is described in terms of the protocol that it is chosen to run. When an FCC is programmed to a certain protocol, it implements a certain level of functionality associated with that protocol. For most protocols, this corresponds to portions of the link layer (layer 2 of the seven-layer OSI model). Many functions of the FCC are common to all of the protocols. These functions are described in the FCC description. Following that, the implementation details that differentiate protocols from one another are discussed, beginning with the transparent protocol. Thus, the reader should read from this point to the transparent protocol and then skip to the appropriate protocol. Since the FCCs use similar data structures across all protocols, the reader's learning time decreases dramatically after understanding the first protocol.

Each FCC supports a number of protocols—Ethernet, HDLC/SDLC, ATM, and totally transparent operation. Although the selected protocol usually applies to both the FCC transmitter and receiver, half of one FCC can run transparent operation while the other runs HDLC/SDLC protocol. The internal clocks (RCLK, TCLK) for each FCC can be programmed with either an external or internal source. The internal clocks originate from one of the baud-rate generators or one of the external clock signals. These clocks can be as fast as one-third the CPM clock frequency. See Chapter 22, “Serial Interface with Time-Slot Assigner.” However, the FCC’s ability to support a sustained bit stream depends on the protocol as well as on other factors. Each FCC can be connected to its own set of pins on the MPC8560. This configuration, the nonmultiplexed serial interface, or NMSI, is described in Chapter 22, “Serial Interface with Time-Slot Assigner.” In this configuration, each FCC can support the standard modem interface signals ($\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$) through the appropriate port pins and the interrupt controller. Additional handshake signals can be supported with additional parallel I/O lines. The FCC block diagram is shown in Figure 34-1.Figure 34-1.

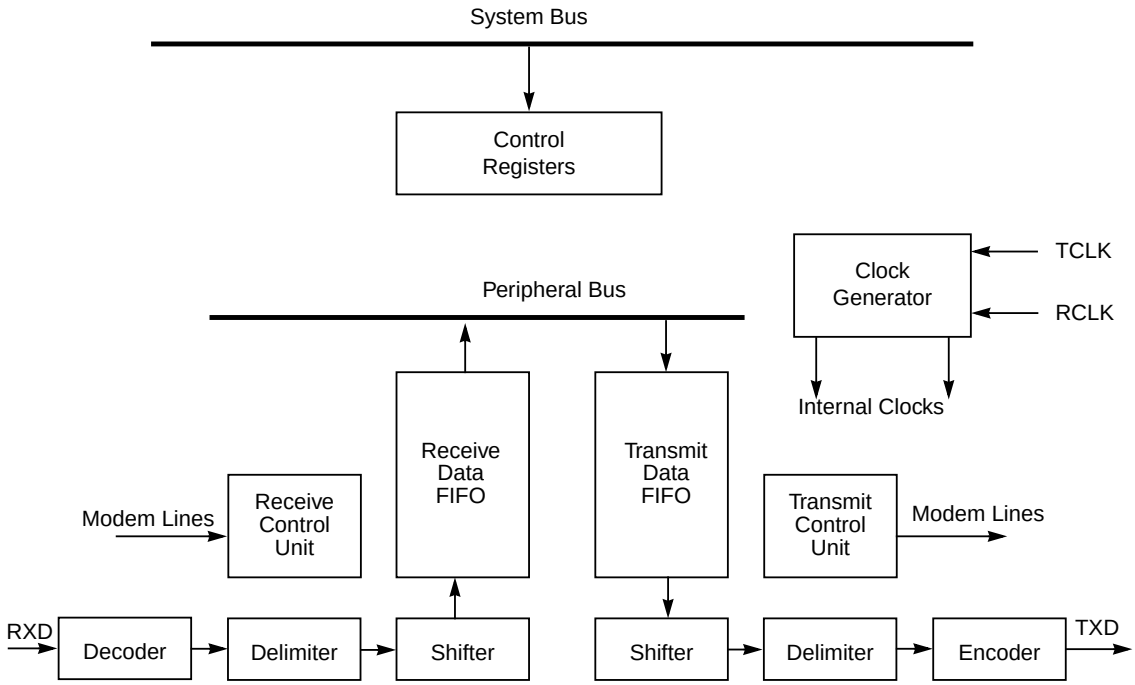


Figure 34-1. FCC Block Diagram

34.2 General FCC Mode Registers (GFMR*x*)

Each FCC contains a general FCC mode register (GFMR*x*) that defines common FCC options and selects the protocol to be run. The GFMR*x* are read/write registers cleared at reset. Figure 34-2 shows the GFMR format.

	0	1	2	3	4	5	6	7	8	9	15
Field	DIAG		TCI	TRX	TTX	CDP	CTSP	CDS	CTSS	—	
Reset	0000_0000_0000_0000										
R/W	R/W										
Addr	0x9_1300 (GFMR1), 0x9_1320 (GFMR2), 0x9_1340 (GFMR3)										

	16	17	18	19	20	21	22	23	24	25	26	27	28	31
Field	SYNL		RTSM	RENC		REVD	TENC		TCRC		ENR	ENT	MODE	
Reset	0000_0000_0000_0000													
R/W	R/W													
Addr	0x9_1302 (GFMR1), 0x9_1322 (GFMR2), 0x9_1342 (GFMR3)													

Figure 34-2. General FCC Mode Register (GFMR)

Table 34-1 describes GFMR fields.

Table 34-1. GFMR Register Field Descriptions

Bits	Name	Description
0–1	DIAG	Diagnostic mode. 00 Normal operation—Receive data enters through RXD, and transmit data is shifted out through TXD. The FCC uses the modem signals ($\overline{CD}$ and $\overline{CTS}$) to automatically enable and disable transmission and reception. Timings are shown in Section 34.12, “FCC Timing Control.” 01 Local loopback mode—Transmitter output is connected internally to the receiver input, while the receiver and the transmitter operate normally. RXD is ignored. Data can be programmed to appear on TXD, or TXD can remain high by programming the appropriate parallel port register. $\overline{RTS}$ can be disabled in the appropriate parallel I/O register. The transmitter and receiver must use the same clock source, but separate CLK*x* pins can be used if connected to the same external clock source. If external loopback is preferred, program DIAG for normal operation and externally connect TXD and RXD. Then, physically connect the control signals ($\overline{RTS}$ connected to $\overline{CD}$, and $\overline{CTS}$ grounded) or set the parallel I/O registers so $\overline{CD}$ and $\overline{CTS}$ are permanently asserted to the FCC by configuring the associated $\overline{CTS}$ and $\overline{CD}$ pins as general-purpose I/O.; see Chapter 42, “Parallel I/O Ports.” 10 Automatic echo mode—The channel automatically retransmits received data, using the receive clock provided. The receiver operates normally and receives data if $\overline{CD}$ is asserted. The transmitter simply transmits received data. In this mode, $\overline{CTS}$ is ignored. The echo function can also be accomplished in software by receiving buffers from an FCC, linking them to TxBDs, and transmitting them back out of that FCC. 11 Loopback and echo mode—Loopback and echo operation occur simultaneously. $\overline{CD}$ and $\overline{CTS}$ are ignored. Refer to the loopback bit description for clocking requirements. For TDM operation, the diagnostic mode is selected by SI*x*MR[SDM*x*]; see Section 22.5.2, “SI Mode Registers (SIxMR).”
2	TCI	Transmit clock invert 0 Normal operation. 1 The FCC inverts the internal transmit clock. The edge on which the FCC outputs the data depends on the protocol: • In HDLC and Transparent mode, when TCI=0, data is sent on the falling edge; when TCI=1, on the rising edge . • In Etherent mode, when TCI=0, data is sent on the rising edge; when TCI=1, on the falling edge.
3	TRX	Transparent receiver. The MPC8560 FCCs offer totally transparent operation. However, to increase flexibility, totally transparent operation is configured with the TTX and TRX bits instead of the MODE bits. This lets the user implement unique applications such as an FCC transmitter configured to HDLC and a receiver configured to totally transparent operation. To do this, program MODE = HDLC, TTX = 0, and TRX = 1. 0 Normal operation 1 The receiver operates in totally transparent mode, regardless of the protocol selected for the transmitter in the MODE bits. Note that full-duplex, totally transparent operation for an FCC is obtained by setting both TTX and TRX. Attempting to operate an FCC with Ethernet or ATM on its transmitter and transparent operation on its receiver causes erratic behavior. In other words, if the MODE = Ethernet or ATM, TTX must equal TRX.

Table 34-1. GFMR Register Field Descriptions (continued)

Bits	Name	Description
4	TTX	Transparent transmitter. The MPC8560 FCCs offer totally transparent operation. However, to increase flexibility, totally transparent operation is configured with the TTX and TRX bits instead of the MODE bits. This lets the user implement unique applications, such as configuring an FCC receiver to HDLC and a transmitter to totally transparent operation. To do this, program MODE = HDLC, TTX = 1, and TRX = 0. 0 Normal operation. 1 The transmitter operates in totally transparent mode, regardless of the receiver protocol selected in the MODE bits. Note that full-duplex totally transparent operation for an FCC is obtained by setting both TTX and TRX. Attempting to operate an FCC with Ethernet or ATM on its receiver and transparent operation on its transmitter causes erratic behavior. In other words, if GFMR[MODE] selects Ethernet or ATM, TTX must equal TRX.
5	CDP	$\overline{CD}$ pulse (transparent mode only) 0 Normal operation (envelope mode). $\overline{CD}$ should envelope the frame; to negate $\overline{CD}$ while receiving causes a $\overline{CD}$ lost error. 1 Pulse mode. Once $\overline{CD}$ is asserted (high to low transition), synchronization has been achieved, and further transitions of $\overline{CD}$ do not affect reception. CDP must be set if this FCC is used with the TSA in transparent mode.
6	CTSP	$\overline{CTS}$ pulse 0 Normal operation (envelope mode). $\overline{CTS}$ should envelope the frame; to negate $\overline{CTS}$ while transmitting causes a $\overline{CTS}$ lost error. See Section 34.12, “FCC Timing Control.” 1 Pulse mode. $\overline{CTS}$ is asserted when synchronization is achieved; further transitions of $\overline{CTS}$ do not affect transmission. When running HDLC, the FCC samples $\overline{CTS}$ only once before sending the first frame after the transmitter is enabled (ENT = 1).
7	CDS	$\overline{CD}$ sampling 0 The $\overline{CD}$ input is assumed to be asynchronous with the data. The FCC synchronizes it internally before data is received. (This mode is not allowed in transparent mode when SYNL = 0b00.) 1 The $\overline{CD}$ input is assumed to be synchronous with the data, giving faster operation. In this mode, $\overline{CD}$ must transition while the receive clock is in the low state. When $\overline{CD}$ goes low, data is received. This is useful when connecting MPC8560s in transparent mode since it allows the $\overline{RTS}$ signal of one MPC8560 to be connected directly to the $\overline{CD}$ signal of another MPC8560.
8	CTSS	$\overline{CTS}$ sampling 0 The $\overline{CTS}$ input is assumed to be asynchronous with the data. When it is internally synchronized by the FCC, data is sent after a delay of no more than two serial clocks. 1 The $\overline{CTS}$ input is assumed to be synchronous with the data, giving faster operation. In this mode, $\overline{CTS}$ must transition while the transmit clock is in the low state. As soon as $\overline{CTS}$ is low, data transmission begins. This mode is useful when connecting MPC8560 in transparent mode because it allows the $\overline{RTS}$ signal of one MPC8560 to be connected directly to the $\overline{CTS}$ signal of another MPC8560.
9--15	—	Reserved, should be 0.
16–17	SYNL	Sync length (transparent mode only). Determines the operation of an FCC receiver configured for totally transparent operation only. See Section 39.3.1, “In-Line Synchronization Pattern.” 00 The sync pattern in the FDSR is not used. An external sync signal is used instead ($\overline{CD}$ signal asserted: high to low transition). 01 Automatic sync (assumes always synchronized, ignores $\overline{CD}$ signal). 10 8-bit sync. The receiver synchronizes on an 8-bit sync pattern stored in the FDSR. Negation of $\overline{CD}$ causes CD lost error. 11 16-bit sync. The receiver synchronizes on a 16-bit sync pattern stored in the FDSR. Negation of $\overline{CD}$ causes CD lost error. Note that if SYNL = 1x, CDP should be cleared (not in $\overline{CD}$ pulse mode).

Table 34-1. GFMR Register Field Descriptions (continued)

Bits	Name	Description
18	RTSM	$\overline{\text{RTS}}$ mode 0 Send idles between frames as defined by the protocol. $\overline{\text{RTS}}$ is negated between frames (default). 1 Send flags/syncs between frames according to the protocol. $\overline{\text{RTS}}$ is asserted whenever the FCC is enabled.
19–20	RENC	Receiver decoding method. The user should set RENC = TENC in most applications. 00 NRZ 01 NRZI (one bit mode HDLC or transparent only) 1x Reserved
21	REVD	Reverse data (valid for a totally transparent channel only) 0 Normal operation 1 The totally transparent channels on this FCC (either the receiver, transmitter, or both, as defined by TTX and TRX) reverse bit order, transmitting the MSB of each octet first.
22–23	TENC	Transmitter encoding method. The user should set TENC = RENC in most applications. 00 NRZ 01 NRZI (one bit mode HDLC or transparent only) 1x Reserved
24-25	TCRC	Transparent CRC (totally transparent channel only). Selects the type of frame checking provided on the transparent channels of the FCC (either the receiver, transmitter, or both, as defined by TTX and TRX). This configuration selects a frame check type; the decision to send the frame check is made in the TxBD. Thus, it is not required to send a frame check in transparent mode. If a frame check is not used, the user can ignore any frame check errors generated on the receiver. 00 16-bit CCITT CRC (HDLC). (X16 + X12 + X5 + 1) 01 Reserved 10 32-bit CCITT CRC (Ethernet and HDLC) (X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1) 11 Reserved
26	ENR	Enable receive. Enables the receiver hardware state machine for this FCC. 0 The receiver is disabled and any data in the receive FIFO buffer is lost. If ENR is cleared during reception, the receiver aborts the current character. 1 The receiver is enabled. ENR may be set or cleared regardless of whether serial clocks are present. Describes how to disable and reenable an FCC. Note that the FCC provides other tools for controlling reception—the ENTER HUNT MODE command, CLOSE RXBD command, and RxBD[E].

Table 34-1. GFMR Register Field Descriptions (continued)

Bits	Name	Description
27	ENT	Enable transmit. Enables the transmitter hardware state machine for this FCC. 0 The transmitter is disabled. If ENT is cleared during transmission, the transmitter aborts the current character and TXD returns to idle state. Data in the transmit shift register is not sent. 1 The transmitter is enabled. ENT can be set or cleared, regardless of whether serial clocks are present. See Section 34.13, “Disabling the FCCs On the Fly,” for a description of the proper methods to disable and reenable an FCC. Note that the FCC provides other tools for controlling transmission besides the ENT bit—the STOP TRANSMIT, GRACEFUL STOP TRANSMIT, and RESTART TRANSMIT commands, $\overline{\text{CTS}}$ flow control, and TxBD[R].
28–31	MODE	Channel protocol mode 0000 HDLC 0001 Reserved for RAM microcode 0010 Reserved 0011 Reserved for RAM microcode 0100 Reserved 0101 Reserved for RAM microcode 0110 Reserved 0111 Reserved for RAM microcode 1000 Reserved 1001 Reserved for RAM microcode 1010 ATM 1011 Reserved for RAM microcode 1100 Ethernet 11xx Reserved

34.3 General FCC Expansion Mode Register (GFEMR)

The general FCC expansion mode register (GFEMR) defines the expansion modes. It should be programmed according to the protocol used.

	0	1	2	3 … 7
Field	TIREM	LPB	CLK	—
Reset	0000_0000			
R/W	R/W			
Addr	0x9_1390 (GFEMR1), 0x9_13B0(GFEMR2), 0x9_13D0(GFEMR3)			

Figure 34-3. General FCC Expansion Mode Register (GFEMR)

Table 34-2 describes GFEMR*x* fields.

Table 34-2. GFEMR*x* Field Descriptions

Bit	Name	Description
0	TIREM	Transmit internal rate expanded mode (ATM mode) 0 Internal rate mode: Internal rate for PHYs[0-3] is controlled only by FTIRR[0-3]. FIRPER, FIRSR_HI, FIRSR_LO, FITER are unused. 1 Internal rate expanded mode: PHYs[0-31] are controlled by FTIRR[0-3], FIRPER, FIRSR_HI and FIRSR_LO. Underrun status for PHYs[0-31] is available by FIRER. This bit should be set only in transmit master multi-PHY mode. In this mode mixing of internal rate and external rate is not enabled.
1	LPB	RMII Loopback diagnostic mode (Ethernet mode): 0 Normal mode 1 Loopback mode
2	CLK	RMII reference clock rate for 50 Mhz input clock from external oscillator (Ethernet mode): 0 50 Mhz (for Fast Ethernet) 1 5 Mhz (for 10BaseT)
3–7	—	Reserved, should be cleared.

34.4 FCC Protocol-Specific Mode Registers (FPSMR*x*)

The functionality of the FCC varies according to the protocol selected by GFMR[MODE]. Each FCC has an additional 32-bit, memory-mapped, read/write protocol-specific mode register (FPSMR) that configures them specifically for a chosen mode. The section for each specific protocol describes the FPSMR bits.

34.5 FCC Data Synchronization Registers (FDSR*x*)

Each FCC has a 16-bit, memory-mapped, read/write data synchronization register (FDSR), shown in Figure 34-2, that specifies the pattern used in the frame synchronization procedure of the synchronous protocols. In the totally transparent protocol, the FDSR should be programmed with the preferred SYNC pattern. For Ethernet protocol, it should be programmed with 0xD555. At reset, it defaults to 0x7E7E (two HDLC flags), so it does not need to be written for HDLC mode. The FDSR contents are always sent lsb first.

	0							7	8							15
Field	SYN2								SYN1							
Reset	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0
R/W	R/W															
Address	0x9_130C (FDSR1), 0x9_132C (FDSR2), 0x9_134C (FDSR3)															

Table 34-3. FCC Data Synchronization Register (FDSR)

34.6 FCC Transmit-on-Demand Registers (FTODR*x*)

If no frame is being sent by the FCC, the CP periodically polls the R bit of the next TxBD to see if the user has requested a new frame/buffer to be sent. The polling algorithm depends on the FCC configuration, but occurs every 256 serial transmit clocks. The user, however, can request that the CP begin processing the new frame/buffer without waiting the normal polling time. For immediate processing, set the transmit-on-demand (TOD) bit in the transmit-on-demand register (FTODR), shown in Figure 34-4, after setting TxBD[R].

This feature, which decreases the transmission latency of the transmit buffer/frame, is particularly useful in LAN-type protocols where maximum interframe GAP times are limited by the protocol specification. Since the transmit-on-demand feature gives a high priority to the specified TxBD, it can conceivably affect the servicing of the other FCC FIFO buffers. Therefore, it is recommended that the transmit-on-demand feature be used only for a high-priority TxBD and when transmission on this FCC has not occurred for a given time period, which is protocol-dependent.

If a new TxBD is added to the BD table while preceding TxBDs have not completed transmission, the new TxBD is processed immediately after the older TxBDs are sent.

	0	1 … 15
Field	TOD	—
Reset	0000_0000_0000_0000	
R/W	R/W	
Addr	0x9_1308 (FTODR1), 0x9_1328 (FTODR2), 0x9_1348 (FTODR3)	

Figure 34-4. FCC Transmit-on-Demand Register (FTODR)

Fields in the FTODR are described in Table 34-4.

Table 34-4. FTODR Field Descriptions

Field	Name	Description
0	TOD	Transmit on demand 0 Normal polling. 1 The CP gives high priority to the current TxBD and begins sending the frame does without waiting for the normal polling time to check TxBD[R]. TOD is cleared automatically.
1–15	—	Reserved, should be cleared.

34.7 FCC Buffer Descriptors

Data associated with each FCC is stored in buffers. Each buffer is referenced by a buffer descriptor (BD). All of the transmit BDs for an FCC are grouped into a TxBD circular table with a programmable length. Likewise, receive BDs form an RxBD table. The user can program the start address of the BD tables anywhere in system memory. See Figure 34-5.

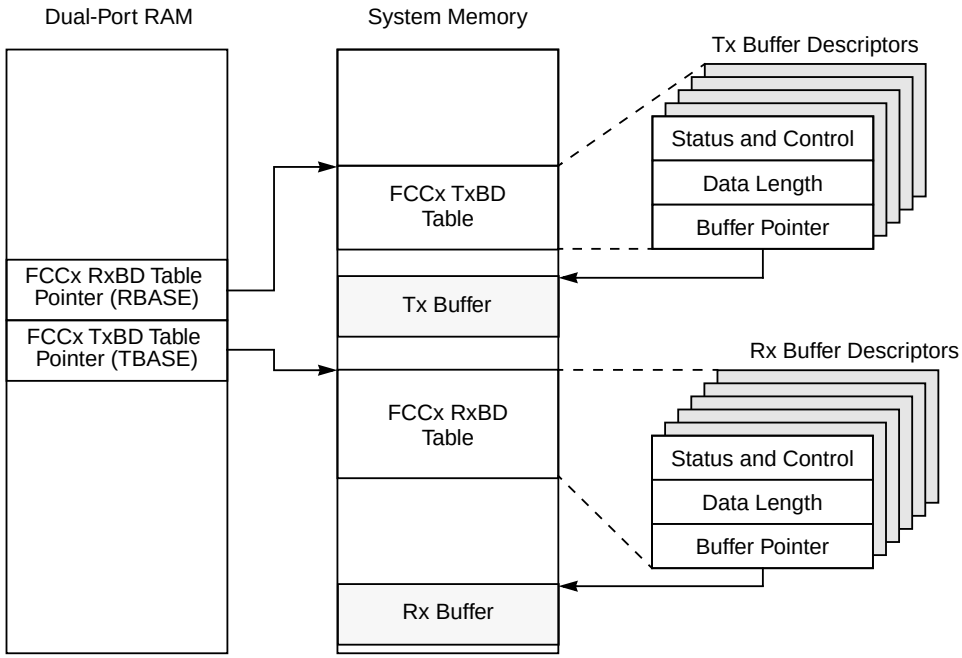


Figure 34-5. FCC Memory Structure

The format of transmit and receive BDs, shown in Figure 34-6, is the same for every FCC mode of operation except ATM mode; see Section 35.10.5, “ATM Controller Buffer Descriptors (BDs).” The first 16 bits in each BD contain status and control information, which differs for each protocol. The second 16 bits indicate the data buffer length in bytes (the wrap bit is the BD table length indicator). The remaining 32-bits contain the 32-bit address pointer to the actual buffer in memory.

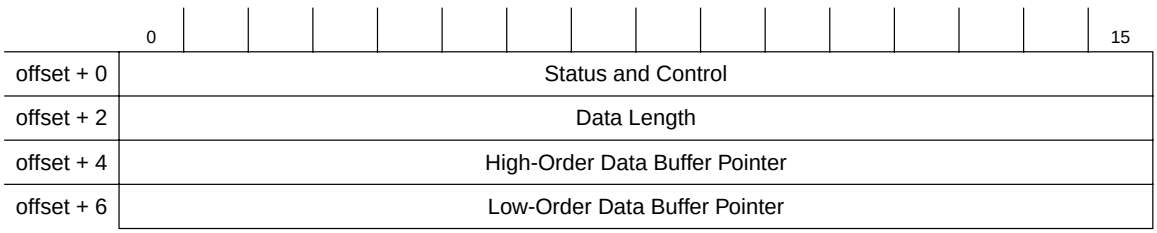


Figure 34-6. Buffer Descriptor Format

For frame-based protocols, a message can reside in as many buffers as necessary (transmit or receive). Each buffer has a maximum length of (64K–1) bytes. The CP does not assume that all buffers of a single frame are currently linked to the BD table. It does assume, however, that unlinked buffers are provided by the core soon enough to be sent or received. Failure to do so causes an error condition being reported by the CP. An underrun error is reported in the case of transmit; a busy error is reported in the case of receive. Because BDs are prefetched, the receive BD table must always contain at least one empty BD to avoid a busy error; therefore, RxBD tables must always have at least two BDs.

The BDs and data buffers can be anywhere in the system memory.

The CP processes the TxBDs in a straightforward fashion. Once the transmit side of an FCC is enabled, it starts with the first BD in that FCC's TxBD table. When the CP detects that TxBD[R] is set, it begins processing the buffer. The CP detects that the BD is ready either by polling the R bit periodically or by the user writing to the FTODR. When the data from the BD has been placed in the transmit FIFO buffer, the CP moves on to the next BD, again waiting for the R bit to be set. Thus, the CP does no look-ahead BD processing, nor does it skip over BDs that are not ready. When the CP sees the wrap (W) bit set in a BD, it goes back to the beginning of the BD table after processing of the BD is complete.

After using a BD, the CP normally clears R (not-ready); thus, the CP does not use a BD again until the BD has been prepared by the core. Some protocols support continuous mode, which allows repeated transmission and for which the R bit remains set (always ready).

The CP uses RxBDs in a similar fashion. Once the receive side of an FCC is enabled, it starts with the first BD in the FCC's RxBD table. Once data arrives from the serial line into the FCC, the CP performs the required protocol processing on the data and moves the resultant data to the buffer pointed to by the first BD. Use of a BD is complete when no room is left in the buffer or when certain events occur, such as the detection of an error or end-of-frame. Regardless of the reason, the buffer is then said to be closed and additional data is stored using the next BD. Whenever the CP needs to begin using a BD because new data is arriving, it checks the E bit of that BD. This check is made on a prefetched copy of the current BD. If the current BD is not empty, it reports a busy error. However, it does not move from the current BD until it is empty. Because there is a periodic prefetch of the RxBD, the busy error may recur if the BD is not prepared soon enough.

When the CP sees the W bit set in a BD, it returns to the beginning of the BD table after processing of the BD is complete. After using a BD, the CP clears the E bit (not empty) and does not use a BD again until the BD has been processed by the core. However, in continuous mode, available to some protocols, the E bit remains set (always empty).

34.8 FCC Parameter RAM

Each FCC parameter RAM area begins at the same offset from each FCC base area. The protocol-specific portions of the FCC parameter RAM are discussed in the specific protocol descriptions. Table 34-5 shows portions common to all FCC protocols.

Some parameter RAM values must be initialized before the FCC is enabled; other values are initialized/written by the CP. Once initialized, most parameter RAM values do not need to be accessed by user software because most activity centers around the TxBDs and RxBDs rather than the parameter RAM. However, if the parameter RAM is accessed, note the following:

- Parameter RAM can be read at any time.

- Tx parameter RAM can be written only when the transmitter is disabled—after a STOP TRANSMIT command and before a RESTART TRANSMIT command or after the buffer/frame finishes transmitting after a GRACEFUL STOP TRANSMIT command and before a RESTART TRANSMIT command.
- Rx parameter RAM can be written only when the receiver is disabled. Note the CLOSE RXBD command does not stop reception, but it does allow the user to extract data from a partially full Rx buffer.
- See Section 34.13, “Disabling the FCCs On the Fly.”

Some parameters in Table 34-5 are not described and are listed only to provide information for experienced users and for debugging. The user need not access these parameters in normal operation.

Table 34-5. FCC Parameter RAM Common to All Protocols except ATM

Offset [1]	Name	Width	Description
0x00	RIPTR	Hword	Receive internal temporary data pointer. Used by microcode as a temporary buffer for data. Must be 32-byte aligned and the size of the internal buffer must be 32 bytes unless it is stated otherwise in the protocol specification. For best performance, it should be located in the following address ranges: 0x3000–0x4000 or 0xB000–0xC000.
0x02	TIPTR	Hword	Transmit internal temporary data pointer. Used by microcode as a temporary buffer for data. Must be 32-byte aligned and the size of the internal buffer must be 32 bytes unless it is stated otherwise in the protocol specification. For best performance it should be located in the following address ranges: 0x3000–0x4000 or 0xB000–0xC000.
0x04	—	Hword	Reserved, should be cleared.
0x06	MRBLR	Hword	Maximum receive buffer length (a multiple of 32 for all modes). The number of bytes that the FCC receiver writes to a receive buffer before moving to the next buffer. The receiver can write fewer bytes to the buffer than MRBLR if a condition such as an error or end-of-frame occurs, but it never exceeds the MRBLR value. Therefore, user-supplied buffers should be at least as large as the MRBLR. Note that FCC 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 an FCC is operating. Change MRBLR only when the FCC receiver is disabled.
0x08	RSTATE	Word	Receive internal state. The high byte, RSTATE[0–7], contains the function code register; see Section 34.8.1, “FCC Function Code Registers (FCRx).” RSTATE[8–31] is used by the CP and must be cleared initially.
0x0C	RBASE	Word	RxBD base address (must be divisible by eight). Defines the starting location in the memory map for the FCC RxBDs. This provides great flexibility in how FCC RxBDs are partitioned. By selecting RBASE entries for all FCCs and by setting the W bit in the last BD in each BD table, the user can select how many BDs to allocate for the receive side of every FCC. The user must initialize RBASE before enabling the corresponding channel. Furthermore, the user should not configure BD tables of two enabled FCCs to overlap or erratic operation occurs.
0x10	RBDSTAT	Hword	RxBD status and control. Reserved for CP use only.
0x12	RBDLEN	Hword	RxBD data length. A down-count value initialized by the CP with MRBLR and decremented with every byte written by the SDMA channels.

Table 34-5. FCC Parameter RAM Common to All Protocols except ATM (continued)

Offset [1]	Name	Width	Description
0x14	RDPTR	Word	RxBD data pointer. Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x18	TSTATE	Word	Tx internal state. The high byte, TSTATE[0–7], contains the function code register; see Section 34.8.1, “FCC Function Code Registers (FCRx).” TSTATE[8–31] is used by the CP and must be cleared initially.
0x1C	TBASE	Word	TxBD base address (must be divisible by eight). Defines the starting location in the memory map for the FCC TxBDs. This provides great flexibility in how FCC TxBDs are partitioned. By selecting TBASE entries for all FCCs and by setting the W bit in the last BD in each BD table, the user can select how many BDs to allocate for the transmit side of every FCC. The user must initialize TBASE before enabling the corresponding channel. Furthermore, the user should not configure BD tables of two enabled FCCs to overlap or erratic operation occurs.
0x20	TBDSTAT	Hword	TxBD status and control. Reserved for CP use only.
0x22	TBDLEN	Hword	TxBD data length. A down-count value initialized with the TxBD data length and decremented with every byte read by the SDMA channels.
0x24	TDPTR	Word	TxBD data pointer. Updated by the SDMA channels to show the next address in the buffer to be accessed.
0x28	RBPTR	Word	RxBD pointer. Points to the next BD that the receiver transfers data to when it is in idle state or to the current BD during frame processing. After a reset or when the end of the BD table is reached, the CP sets RBPTR = RBASE. Although the user need never write to RBPTR in most applications, the user can modify it when the receiver is disabled or when no receive buffer is in use.
0x2C	TBPTR	Word	TxBD pointer. Points either to the next BD that the transmitter transfers data from when it is in idle state or to the current BD during frame transmission. After a reset or when the end of the BD table is reached, the CP sets TBPTR = TBASE. Although the user need never write to TBPTR in most applications, the user can modify it when the transmitter is disabled or when no transmit buffer is in use (after a STOP TRANSMIT or GRACEFUL STOP TRANSMIT command is issued and the frame completes transmission).
0x30	RCRC	Word	Temporary receive CRC
0x34	—	Word	Reserved
0x38	TCRC	Word	Temporary transmit CRC
0x3C	—	Word	First word of protocol-specific area

[1] Offset from FCC base: 0x8400 (FCC1), 0x8500 (FCC2) and 0x8600 (FCC3); see Section 20.5.2, “Parameter RAM.”

34.8.1 FCC Function Code Registers (FCR*x*)

The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. Figure 34-7 shows the format of the transmit and receive function code registers, which reside at TSTATE[0–7] and RSTATE[0–7] in the FCC parameter RAM.

	0	1	2	3	4	5	6	7
Field	—	FCCP	GBL	BO		—	DTB	BDB

Figure 34-7. Function Code Register (FCR*x*)

FCR*x* fields are described in Table 34-6.

Table 34-6. FCR*x* Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared.
1	FCCP	FCC priority. Used in conjunction with PPC_ACR[PRKM] and LCL_ACR[PRKM] for a low request level. 0 Disables CPM low request level to refer to FCCs and MCCs. 1 Enables CPM low request level to refer to FCCs and MCCs.
2	GBL	Global. Indicates whether the memory operation should be snooped. 0 Snooping disabled. 1 Snooping enabled.
3–4	BO	Byte ordering.Used to select the byte ordering of the buffer. If BO is modified on-the-fly, it takes effect at the start of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD. 0x Reserved 10 1x Big-endian
5	—	Reserved, should be cleared.
6	DTB	Indicates on what bus the data is located. 0 On the system bus. 1 On the local.
7	BDB	Indicates on what bus the BDs are located. 0 On the system bus. 1 On the local bus.

34.9 Interrupts from the FCCs

Interrupt handling for each of the FCC channels is configured on a global (per channel) basis in the interrupt pending register (SIPNR_L) and interrupt mask register (SIMR_L). One bit in each register is used to either mask, enable, or report an interrupt in an FCC channel. The interrupt priority between the FCCs is programmable in the CPM interrupt priority register (SCPRR_H). The interrupt vector register (SIVEC) indicates which pending channel has highest priority. Registers within the FCCs manage interrupt handling for FCC-specific events.

Events that can cause the FCC to interrupt the processor vary slightly among protocols and are described with each protocol. These events are handled independently for each channel by the FCC event and mask registers (FCCE and FCCM).

34.9.1 FCC Event Registers (FCCE*x*)

Each FCC has an FCC event register (FCCE) used to report events. On recognition of an event, the FCC sets its corresponding FCCE bit regardless of the corresponding mask bit. To the user it appears as a memory-mapped register that can be read at any time. Bits are cleared by writing ones; writing zeros has no effect on bit values. FCCE is cleared at reset. Fields of this register are protocol-dependent and are described in the respective protocol sections.

34.9.2 FCC Mask Registers (FCCM*x*)

Each FCC has a read/write FCC mask register (FCCM) used to enable or disable CP interrupts to the core for events reported in an event register (FCCE). Bit positions in FCCM are identical to those in FCCE. Note that an interrupt is generated only if the FCC interrupts are also enabled in the SIU; see Section 21.5.1.4, “CPM Interrupt Mask Registers (SIMR_H and SIMR_L).”

If an FCCM bit is zero, the CP does not proceed with its usual interrupt handling whenever that event occurs. Any time a bit in the FCCM register is set, a 1 in the corresponding bit in the FCCE register sets the FCC event bit in the interrupt pending register; see Section 21.5.1.4, “CPM Interrupt Mask Registers (SIMR_H and SIMR_L).”

34.9.3 FCC Status Registers (FCCS*x*)

Each FCC has an 8-bit, read/write FCC status register (FCCS) that lets the user monitor real-time status conditions (flags, idle) on the RXD line. It does not show the status of $\overline{\text{CTS}}$ and $\overline{\text{CD}}$; their real-time status is available in the appropriate parallel I/O port (see Chapter 42, “Parallel I/O Ports”).

34.10 FCC Initialization

The FCCs require a number of registers and parameters to be configured after a power-on reset. The following outline gives the proper sequence for initializing the FCCs, regardless of the protocol used.

1. Write the parallel I/O ports to configure and connect the I/O pins to the FCCs.
2. Write the appropriate port registers to configure $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ to be parallel I/O with interrupt capability or to connect directly to the FCC (if modem support is needed).
3. If the TSA is used, the SI must be configured. If the FCC is used in the NMSI mode, the CPM multiplexing logic (CMX) must still be initialized.
4. Write the GFMR, but do not write the ENT or ENR bits yet.
5. Write the FPSMR.

6. Write the FDSR.
7. Initialize the required values for this FCC in its parameter RAM.
8. Clear out any current events in FCCE, as needed.
9. Write the FCCM register to enable the interrupts in the FCCE register.
10. Write the SCPRR_H to configure the FCC interrupt priority.
11. Clear out any current interrupts in the SIPNR_L, if preferred.
12. Write the SIMR_L to enable interrupts to the CP interrupt controller.
13. Issue an INIT TX AND RX PARAMETERS command (with the correct protocol number).
14. Set GFMR[ENT] and GFMR[ENR].

The first RxBD's empty bit must be set before the INIT RX COMMAND. However TxBDs can have their ready bits set at any time. Notice that the CPCR does not need to be accessed after a power-on reset until an FCC is to be used. An FCC should be disabled and reenabled after any dynamic change in its parallel I/O ports or serial channel physical interface configuration. A full reset using CPCR[RST] is a comprehensive reset that also can be used.

34.11 FCC Interrupt Handling

The following describes what usually occurs within an FCC interrupt handler:

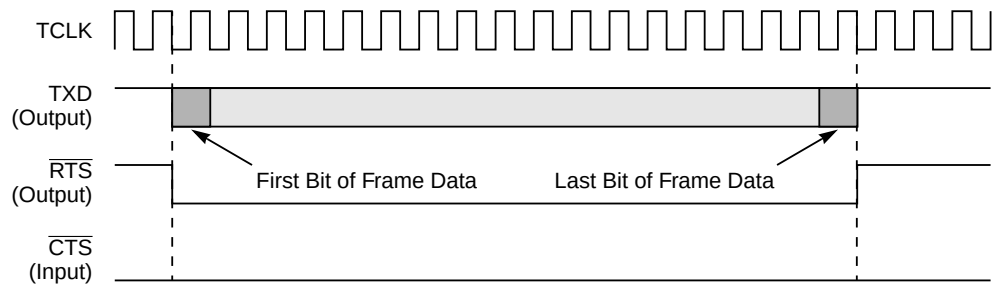
1. When an interrupt occurs, read FCCE to determine interrupt sources. FCCE bits to be handled in this interrupt handler are normally cleared at this time.
2. Process the TxBDs to reuse them if the FCCE[TX,TXE] were set. If the transmit speed is fast or the interrupt delay is long, more than one transmit buffer may have been sent by the FCC. 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.
3. Extract data from the RxBD if FCCE[RX, RXB, or RXF] is set. If the receive speed is fast or the interrupt delay is long, the FCC 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 FCC prefetches BDs, the BD table must be big enough such that always there will be another empty BD to prefetch.
4. Clear FCCE.
5. Continue normal execution.

34.12 FCC Timing Control

When GFMR[DIAG] is programmed to normal operation, $\overline{CD}$ and $\overline{CTS}$ are automatically controlled by the FCC. GFMR[TCI] is assumed to be cleared, which implies normal transmit clock operation.

$\overline{RTS}$ is asserted when FCC has data to transmit in the transmit FIFO and a falling transmit clock occurs. At this point, the FCC begins sending the data, once the appropriate conditions occur on $\overline{CTS}$. In all cases, the first bit of data is the start of the opening flag, or sync pattern.

Figure 34-8 shows that the delay between $\overline{RTS}$ and data is 0 bit times, regardless of the setting of GFMR[CTSS]. This operation assumes that $\overline{CTS}$ is either already asserted to the FCC or is reprogrammed to be a parallel I/O line, in which case the $\overline{CTS}$ signal to the FCC is always asserted. $\overline{RTS}$ is negated one clock after the last bit in the frame.



Note:
1. A frame includes opening and closing flags and syncs, if present in the protocol.

Figure 34-8. Output Delay from $\overline{RTS}$ Asserted

If $\overline{CTS}$ is not already asserted when $\overline{RTS}$ is asserted, the delays to the first bit of data depend on when $\overline{CTS}$ is asserted. Figure 34-9 shows that the delay between $\overline{CTS}$ and the data can be approximately 0.5 to 1 bit times or no delay, depending on GFMR[CTSS].

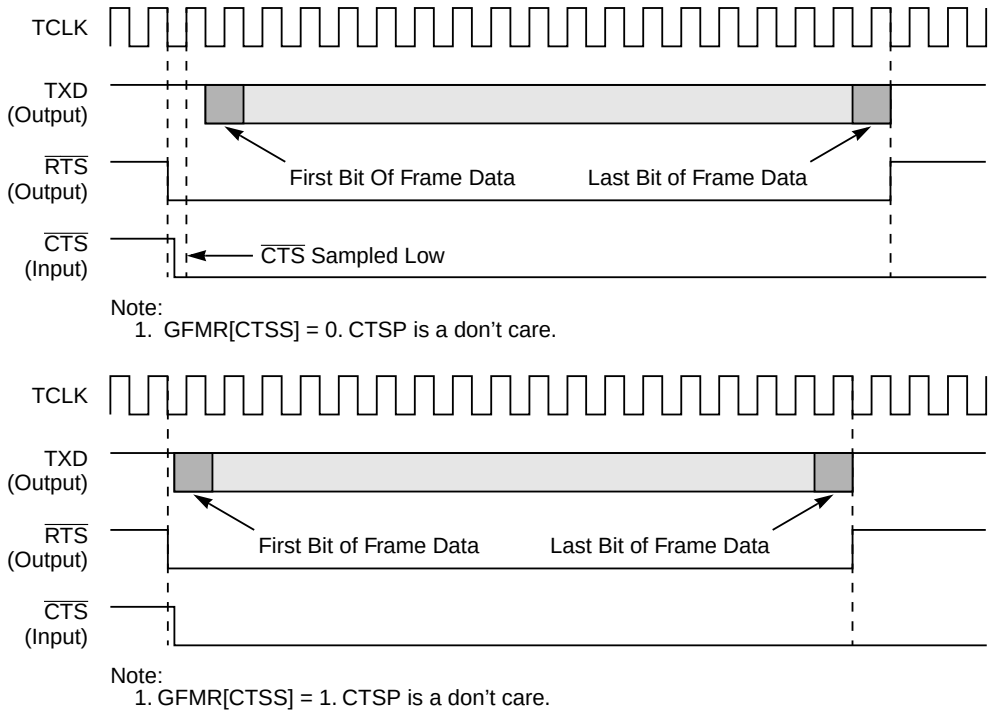


Figure 34-9. Output Delay from $\overline{\text{CTS}}$ Asserted

If it is programmed to envelope the data, $\overline{\text{CTS}}$ must remain asserted during frame transmission or a $\overline{\text{CTS}}$ lost error occurs. The negation of $\overline{\text{CTS}}$ forces $\overline{\text{RTS}}$ high and the transmit data to the idle state. If GFMR[CTSS] = 0, the FCC must sample $\overline{\text{CTS}}$ before a $\overline{\text{CTS}}$ lost is recognized. Otherwise, the negation of $\overline{\text{CTS}}$ immediately causes the $\overline{\text{CTS}}$ lost condition. See Figure 34-10.

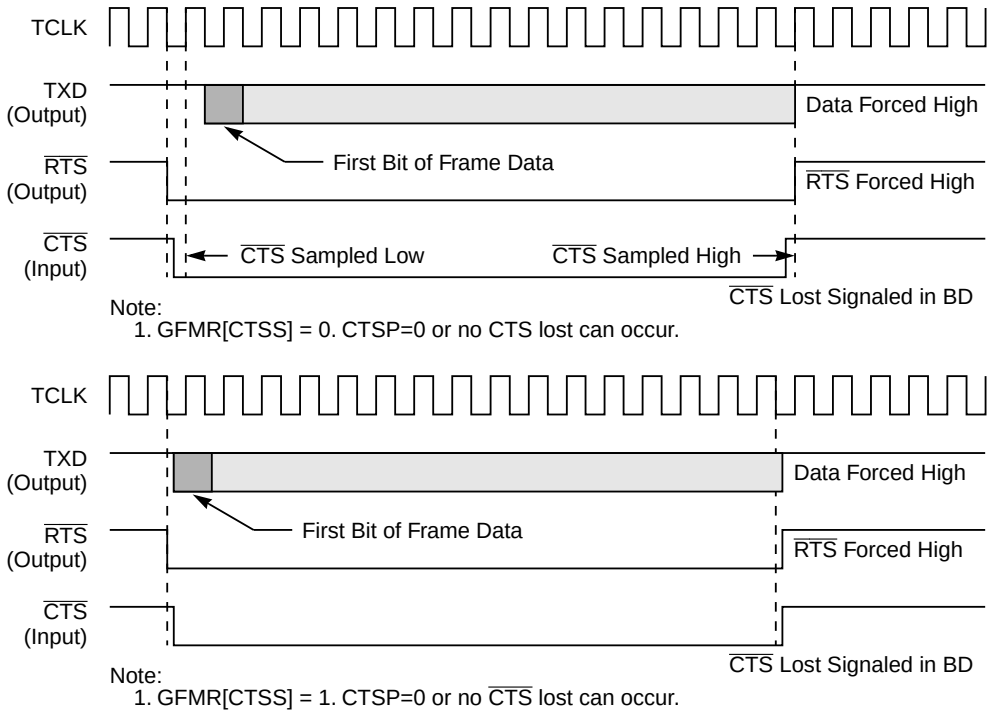


Figure 34-10. $\overline{\text{CTS}}$ Lost

NOTE

Note that if GFMR[CTSS] = 1, all $\overline{\text{CTS}}$ transitions must occur while the transmit clock is low.

Reception delays are determined by $\overline{\text{CD}}$ as Figure 34-11 shows. If GFMR[CDS] = 0, $\overline{\text{CD}}$ is sampled on the rising receive clock edge before data is received. If GFMR[CDS] = 1, $\overline{\text{CD}}$ transitions immediately cause data to be gated into the receiver.

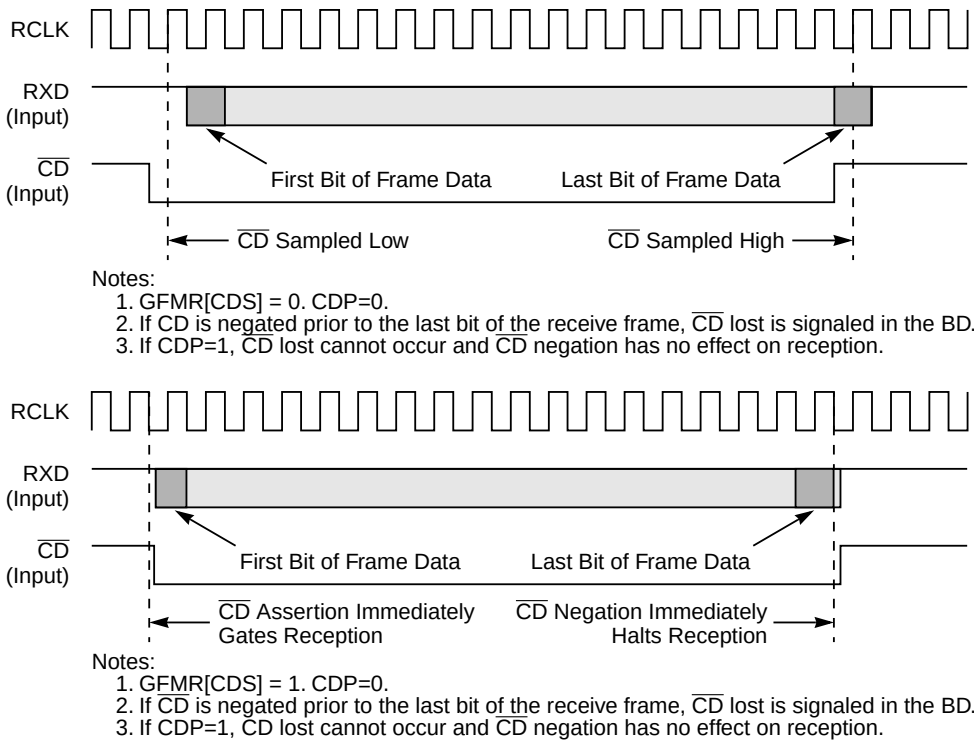


Figure 34-11. Using $\overline{CD}$ to Control Reception

If it is programmed to envelope data, $\overline{CD}$ must remain asserted during frame transmission or a $\overline{CD}$ lost error occurs. The negation of $\overline{CD}$ terminates reception. If [CDS] = 0, $\overline{CD}$ must be sampled by the FCC before a $\overline{CD}$ lost is recognized. Otherwise, the negation of $\overline{CD}$ immediately causes the $\overline{CD}$ lost condition.

Note that if GFMR[CDS] = 1, all $\overline{CD}$ transitions must occur while the receive clock is low.

34.13 Disabling the FCCs On the Fly

Unused FCCs can be temporarily disabled. In this case, a operation sequence is followed that ensures that any buffers in use are closed properly and that new data is transferred to or from a new buffer. Such a sequence is required if the parameters that must be changed are not allowed to be changed dynamically. If the register or bit description states that dynamic changes are allowed, the following sequences are not required and the register or bit may be changed immediately. In all other cases, the sequence should be used.

Modifying parameter RAM does not require the FCC to be fully disabled. See the parameter RAM description for when values can be changed. To disable all peripheral controllers, set CPCR[RST] to reset the entire CPM.

34.13.1FCC Transmitter Full Sequence

For the FCC transmitter, the full disable and enable sequence is as follows.

1. Issue the STOP TRANSMIT command. This is recommended if the FCC is currently transmitting data because it stops transmission in an orderly way. If the FCC is not transmitting (no TxBDs are ready or the GRACEFUL STOP TRANSMIT command has been issued and completed), then the STOP TRANSMIT command is not required. Furthermore, if the TBPTR is overwritten by the user or the INIT TX PARAMETERS command is executed, this command is not required.
2. Clear GFMR[ENT]. This disables the FCC transmitter and puts it in a reset state.
3. Make changes. The user can modify FCC transmit parameters, including the parameter RAM. To switch protocols or restore the FCC transmit parameters to their initial state, the INIT TX PARAMETERS command must be issued.
4. If an INIT TX PARAMETERS command was not issued in step 3, issue a RESTART TRANSMIT command.
5. Set GFMR[ENT]. Transmission begins using the TxBD that the TBPTR points to as soon as TxBD[R] = 1.

34.13.2FCC Transmitter Shortcut Sequence

A shorter sequence is possible if the user prefers to reinitialize the transmit parameters to the state they had after reset. This sequence is as follows:

1. Clear GFMR[ENT].
2. Issue the INIT TX PARAMETERS command. Any additional changes can be made now.
3. Set GFMR[ENT].

34.13.3FCC Receiver Full Sequence

The full disable and enable sequence for the receiver is as follows:

1. Clear GFMR[ENR]. Reception is aborted immediately, which disables the receiver of the FCC and puts it in a reset state.
2. Make changes. The user can modify the FCC receive parameters, including the parameter RAM. If the user prefers to switch protocols or restore the FCC receive parameters to their initial state, the INIT RX PARAMETERS command must be issued.
3. Issue the ENTER HUNT MODE command. This command is required if the INIT RX PARAMETERS command was not issued in step 2.
4. Set GFMR[ENR]. Reception begins immediately using the RxBD that the RBPTR points to if RxBD[E] = 1.

34.13.4 FCC Receiver Shortcut Sequence

A shorter sequence is possible if the user prefers to reinitialize the receive parameters to the state they had after reset. This sequence is as follows:

1. Clear GFMR[ENR].
2. Issue the INIT RX PARAMETERS command. Any additional changes can be made now.
3. Set GFMR[ENR].

34.13.5 Switching Protocols

A user can switch the protocol that the FCC is executing (HDLC) without resetting the board or affecting any other FCC by taking the following steps:

1. Clear GFMR[ENT] and GFMR[ENR].
2. Issue the INIT TX AND RX PARAMETERS command. This command initializes both transmit and receive parameters. Additional changes can be made in the GFMR to change the protocol.
3. Set GFMR[ENT] and GFMR[ENR]. The FCC is enabled with the new protocol.

34.14 Saving Power

Clearing an FCC's ENT and ENR bits minimizes its power consumption.

Chapter 35
ATM Controller

The ATM controller provides the ATM and AAL layers of the ATM protocol using the universal test and operations physical layer (PHY) interface for ATM (UTOPIA level II) for both master and slave modes. It performs segmentation and reassembly (SAR) functions of AAL5, AAL1, and AAL0, and most of the common parts of the convergence sublayer (CP-CS) of these protocols.

NOTE

The bus atomicity mechanism for CAM accesses may not function correctly when the CPM performs a DMA access to an external CAM device. This only impacts systems in which multiple CPMs will access the CAM.

For each virtual channel (VC), the controller's ATM pace control (APC) unit generates a cell transmission rate to implement constant bit rate (CBR), variable bit rate (VBR), available bit rate (ABR), unspecified bit rate (UBR) or UBR+ traffic. To regulate VBR traffic, the APC unit performs a continuous-state leaky bucket algorithm. The APC unit also uses up to eight priority levels to prioritize real-time ATM channels, such as CBR and real-time VBR, over non-real-time ATM channels such as VBR, ABR and UBR.

The ATM controller performs the ATM Forum (UNI-4.0) ABR flow control. To perform feedback rate adaptation, it supports forward and backward resource management (RM) cell generation and ATM Forum floating-point calculation. ABR flow control is implemented in hardware and firmware (without software intervention) to prevent potential delays during backward RM cell processing and feedback rate adaptation.

The MPC8560 supports a special mode for ATM/TDM interworking. The CPM performs automatic data forwarding between ATM channels and the MCCs' TDM channels without core intervention.

The MPC8560 ATM SAR controller applications are as follows:

- ATM line card controllers
- ATM-to-WAN interworking (frame relay, T1/E1 circuit emulation)
- Residential broadband network interface units (NIU) (ATM-to-Ethernet)
- High-performance ATM network interface cards (NIC)

- Bridges and routers with ATM interface

35.1 Features

The ATM controller has the following features:

- Full duplex segmentation and reassembly at 155 Mbps
- UTOPIA level II master and slave modes 8/16 bit
- AAL5, AAL1, AAL0 protocols
- Up to 255 active VCs internally, and up to 64K VCs using external memory
- TM 4.0 CBR, VBR, UBR, UBR+ traffic types
- VBR type 1 and 2 traffic using leaky buckets (GCRA)
- TM 4.0 ABR flow control (EFCI and ER)
- Idle/unassign cells screening/transmission option
- External and internal rate transmit modes
- ATM internal rate mode for 31 PHYs
- ATM 31 PHY addresses for both FCC1 and FCC2
- Special mode for ATM-to-TDM or ATM-to-ATM data forwarding
- CLP and congestion indication marking
- User-defined cells up to 65 bytes
- Separate TxBD and RxBD tables for each virtual channel (VC)
- Special mode of global free buffer pools for dynamic and efficient memory allocation with early packet discard (EPD) support
- Interrupt report per channel using four priority interrupt queues
- Compliant with ATMF UNI 4.0 and ITU specification
- AAL5 cell format
  - — Reassembly
    - – Reassemble PDU directly to external memory
    - – CRC32 check
    - – CLP and congestion report
    - – CPCS_UU, CPI, and length check
    - – Abort message report
  - — Segmentation
    - – Segment PDU directly from external memory
    - – Performs PDU padding
    - – CRC32 generation

  - Automatic last cell marking
  - Automatic CPCS_UU, CPI, and length insertion
  - Abort message option
- AAL1 cell format
  - Reassembly
    - Reassemble PDU directly to external memory
    - Support for partially filled cells (configurable on a per-VC basis)
    - Sequence number check
    - Sequence number protection (CRC-3 and parity) check
  - Segmentation
    - Segment PDU directly from external memory
    - Partially filled cells support (configurable on a per-VC basis)
    - Sequence number generation
    - Sequence number protection (CRC-3 and even parity) generation
  - Structured AAL1 cell format
    - Automatic synchronization using the structured pointer during reassembly
    - Structured pointer generation during segmentation
  - Unstructured AAL1 cell format
    - Clock recovery using external SRTS (synchronous residual time stamp) logic during reassembly
    - SRTS generation using external logic during segmentation
- AAL0 format
  - Receive
    - Whole cell is put in memory
    - CRC10 pass/fail indication
  - Transmit
    - Reads a whole cell from memory
    - CRC10 insertion option
- Support for user-defined cells
  - Support cells up to 65 bytes
  - Extra header insert/load on a per-frame basis
  - Extra header size has byte resolution
  - Asymmetric cell size for send and receive
  - HEC octet insertion option

- PHY
  - UTOPIA level II supports 8/16 bits 25/50 MHz
    - Supports UTOPIA master and slave modes
    - Supports cell-level handshake
    - Supports multiple-PHY polling mode
- ATM pace control (APC) unit
  - Peak cell rate pacing on a per-VC basis
  - Peak-and-sustain cell rate pacing using GCRA on a per-VC basis
  - Peak-and-minimum cell rate pacing on a per-VC basis
  - Up to eight priority levels
  - Fully managed by CP with no host intervention
- Available bit rate (ABR)
  - Performs ATMF UNI 4.0 ABR flow control on a per-VC basis
  - Automatic forward-RM, backward-RM cells generation
  - Automatic feedback rate adaptation
  - Support for EFCI (explicit forward congestion indication) and ER (explicit rate)
  - RM cell floating-point calculations
  - Fully managed by CP with no host intervention
- Receive address look-up mechanism
  - Two modes of address look-up are supported
    - External CAM
    - Address compression
- OAM (operations and maintenance) cells
  - OAM filtering according to PTI field and reserved VCI field
  - Raw cell queues for transmission and reception
  - CRC-10 generation/check
  - Performance monitoring support
    - Support up to 64 bidirectional block tests simultaneously
    - Automatic FMC and BRC cell generation and termination
    - User transmit $cell_{0+1}$ count
    - User transmit $cell_0$ count
    - PM cells time stamp insertion
    - Block error detection code ($BEDC_{0+1}$) generation/check
    - Total receive $cell_{0+1}$ count

  - Total receive $cell_0$ count
  - Specifying channel code for F5 OAM cells
- ATM layer statistic gathering on a per PHY basis.
  - UTOPIA receiver error cells count (Rx parity error or short/long cells error)
  - Misinserted cell count
  - CRC-10 error cells count (ABR flow only)
- Memory management
  - RxBD table per VC with option of global free buffer pool for AAL5
  - TxBD table per VC

35.2 ATM Controller Overview

The following sections provide an overview of the transmitter and receiver portions of the ATM controller.

35.2.1 Transmitter Overview

Before the transmitter is enabled, the host must initialize the MPC8560 and create the transmit data structure, described in Section 35.10, “ATM Memory Structure.” When data is ready for transmission, the host arranges the BD table and writes the pointer of the first BD in the transmit connection table (TCT). The host issues an ATM TRANSMIT command, which inserts the current channel to the ATM pace control (APC) unit. The APC unit controls the ATM traffic of the transmitter. It reads the traffic parameters of each channel and divides the total bandwidth among them. The APC unit can pace the peak cell rate, peak-and-sustain cell rate (GCRA traffic) or peak-and-minimum cell rate traffic. The APC implements up to eight priority levels for servicing real-time channels before non-real-time channels.

The transmitter ATM cell is 53–65 bytes and includes 4 bytes of ATM cell header, a 1-byte HEC, and 48 bytes of payload. The HEC is a constant taken from FDSR*x*[0–15] when using UTOPIA 16 and from FDSR*x*[8–15] when using UTOPIA 8; see Section 34.5, “FCC Data Synchronization Registers (FDSRx).” User-defined cells (UDC mode) include an extra header of 1–12 bytes with an optional HEC octet. Cell transfers use the UTOPIA level II, cell-level handshake.

Transmission starts when the APC schedules a channel. According to the channel code, the ATM controller reads the channel’s entry in the TCT and opens the first BD for transmission. In auto-VC-off mode, the APC automatically deactivates the current channel when no buffer is ready to transmit. In this case, a new ATM TRANSMIT command is needed for transmission of the VC to resume.

35.2.1.1 AAL5 Transmitter Overview

The transmitter reads 48 bytes from the external buffer, adds the cell header, and sends the cell through the UTOPIA interface. The transmitter adds any padding needed and appends the AAL5 trailer in the last cell of the AAL5 frame. The trailer consists of CPCS-UU+CPI, data length, and CRC-32 as defined in ITU I.363. The CPCS-UU+CPI (2-byte entry) can be specified by the user or optionally cleared by the transmitter; see Section 35.10.2.3, “Transmit Connection Table (TCT).” The transmitter identifies the last cell of the AAL5 message by setting the last (L) indication bit in the PTI field of the cell header. An interrupt may be generated to indicate the end of the frame.

When the transmission of the current frame ends and no additional valid buffers are in the BD table, the transmit process ends. The transmitter keeps polling the BD table every time this channel is scheduled to transmit. Note that a buffer-not-ready indication during frame transmission aborts the frame transfer.

35.2.1.2 AAL1 Transmitter Overview

The MPC8560 supports both structured and unstructured AAL1 formats. For the unstructured format, the transmitter reads 47 bytes from the external buffer and inserts them into the AAL1 user data field. The AAL1 PDU header, which consists of the sequence number (SN) and the sequence number protection (SNP) (CRC-3 and parity bit), is generated and inserted into the cell. The MPC8560 supports synchronous residual time stamp (SRTS) generation using external PLL. If this mode is enabled, the MPC8560 reads the SRTS code from the external logic and inserts it into four outgoing cells. See Section 35.15, “SRTS Generation and Clock Recovery Using External Logic.”

For the structured format, the transmitter reads 47 or 46 bytes from the external buffer and inserts them into the AAL1 user data field. The CP generates the AAL1 PDU header and inserts it into the cell. The header consists of the SN, SNP, and the structured pointer.

The MPC8560 supports partially filled cells configured on a per-VC basis. In this mode (TCT[PFM] = 1), only valid octets are copied from the buffer to the ATM cell; the rest of the cell is filled with padding octets.

35.2.1.3 AAL0 Transmitter Overview

No specific adaptation layer is provided for AAL0. The ATM controller reads a whole cell from an external buffer, which always contains exactly one AAL0 cell. The ATM controller optionally generates CRC10 on the cell payload and places it at the end of the payload (CRC10 field). AAL0 mode can be used to send OAM cells or AAL3/4 raw cells.

35.2.1.4 Transmit External Rate and Internal Rate Modes

The ATM controller supports the following three rate modes:

- External rate mode—The total transmission rate is determined by the PHY transmission rate. The FCC sends cells to keep the PHY FIFOs full; the FCC inserts idle/unassign cells to maintain the transmission rate.
- Internal rate mode—The total transmission rate is determined by the FCC internal rate timers. In this mode, the FCC does not insert idle/unassign cells. The internal rate mechanism is supported for the first four PHY devices (PHY address 0-3). Each PHY has its own FTIRR, described in Section 35.13.8, “FCC Transmit Internal Rate Register (FTIRRx).” The FTIRR includes the initial value of the internal rate timer. A cell transmit request is sent when an internal rate timer expires. When using internal rate mode, the user assigns one of the baud-rate generators (BRGs) to clock the four internal rate timers.
- Internal rate expanded mode—The total transmission rate is determined by the FCC internal rate timers and by the assignment of rate per PHY. In this mode, the FCC does not insert idle/unassign cells. The internal rate expanded mode differs from the internal rate mode in that the internal rate mechanism is extended for 31 PHY devices (PHY addresses 0-30) and there cannot be a mix of external and internal rate PHYs. Expanded internal rate is configured by registers GFEMRx, FIRPERx, FIRSRx_HI, FIRSRx_LO, and by FTIRRx. Another feature of internal rate expanded mode is an indication of transmit underrun error status per PHY. When using internal rate expanded mode, the user assigns one of the baud-rate generators (BRGs) to clock the four internal rate timers, and any timer can trigger any PHY.

35.2.2 Receiver Overview

Before the receiver is enabled, the host must initialize the MPC8560 and create the receive data structure described in Section 35.10, “ATM Memory Structure.” The host arranges a BD table for each ATM channel. Buffers for each connection can be statically allocated (that is, each BD in the BD table is associated with a fixed buffer location) or in the case of AAL5, can be fetched by the CP from a global free buffer pool. See Section 35.10.5, “ATM Controller Buffer Descriptors (BDs).”

The receiver ATM cell size is 53-65 bytes. The cell includes: 4 bytes ATM cell header, 1 byte HEC, which is ignored, and 48 bytes payload. User-defined cells (UDC mode) include an extra header of 1–12 bytes with an optional HEC octet. Cell transfers use the UTOPIA level II, cell-level handshake.

Reception starts when the PHY asserts the receive cell available signal (RxCLAV) to indicate that the PHY has a complete cell in its receive FIFO. The receiver reads a complete cell from the UTOPIA interface and translates the header address (VP/VC) to a channel code by performing an address look-up. If no matches are found, the cell is discarded and the user-network interface (UNI) statistics tables are updated. The receiver uses the channel code to read the channel parameters from the receive connection table (RCT).

35.2.2.1 AAL5 Receiver Overview

The receiver copies the 48-byte cell payload to the external buffer and calculates CRC-32 on the entire CPCS-PDU. When the last AAL5 cell arrives, the receiver checks the length, CRC-32, and CPCS-UU+CPI fields and sets the corresponding RxBD status bits. An interrupt may be generated to one of the four interrupt queues. The receiver copies the last cell to memory including the padding and the AAL5 trailer. The CPCS-UU+CPI (16-bit entry) may be read directly from the AAL5 trailer.

The ATM controller monitors the CLP and CNG state of the incoming cells. When the message is closed, these events set RxBD[CLP] and RxBD[CNG].

When no buffer is ready to receive cells (busy state), the receiver switches to hunt state and drops all cells associated with the current frame (partial packet discard). The receiver tries to open new buffers for cell reception only after the last cell of the discarded AAL5 frame arrives.

35.2.2.2 AAL1 Receiver Overview

The ATM controller supports both AAL1 structured and unstructured formats. For the unstructured format, 47 octets are copied to the current receive buffer. The AAL1 PDU header, which consists of the sequence number (SN) and the sequence number protection (SNP) (CRC-3 and parity bit), is checked. The MPC8560 supports SRTS clock recovery using an external PLL. In this mode, the MPC8560 tracks the SRTS from the four incoming cells and writes the SRTS code to external logic. See Section 35.15, “SRTS Generation and Clock Recovery Using External Logic.”

In the unstructured format, when the receive process begins, the receiver hunts for the first cell with a valid sequence number (SN field). When one arrives, the receiver leaves the hunt state and starts receiving. If an SN mismatch is detected, the receiver closes the RxBD, sets the sequence number error flag (RxBD[SNE]), and switches to hunt state, where it stays until a cell with a valid SN field is received.

For the structured format, 47 or 46 octets are copied to the current receive buffer. The AAL1 PDU header, which consists of SN and SNP, is checked and the PDU status is written to the BD.

In the structured format, when the receive process begins, the receiver hunts for the first cell with a valid structured pointer to gain synchronization. When one arrives, the receiver leaves the hunt state and starts receiving. Then the receiver opens a new buffer. The structured pointer points to the first octet of the structured block, which then becomes the first byte of the new buffer. If an SN mismatch is detected, the ATM receiver closes the current RxBD, sets RxBD[SNE], and returns to the hunt state. The receiver then waits for a cell with a valid structured pointer to regain synchronization.

The MPC8560 supports partially filled cells configured on a per-VC basis. In this mode (RCT[PFM] = 1), the ATM controller copies only the valid octets from the cell user data field to the buffer.

35.2.2.3 AAL0 Receiver Overview

For AAL0, no specific adaptation layer processing is done. The ATM controller copies the whole cell to an external buffer. Each buffer contains exactly one AAL0 cell. The ATM controller calculates and checks the CRC10 of the cell payload and sets RxBD[CRE] if a CRC error occurs. AAL0 mode can be useful for receiving OAM cells or AAL3/4 raw cells.

35.2.3 Performance Monitoring

The ATM controller supports performance monitoring testing according to ITU I.610. When performance monitoring is enabled, the ATM controller automatically generates and terminates FMCs (forward monitoring cells) and BRCs (backward reporting cells). See Section 35.6.6, “Performance Monitoring.”

35.2.4 ABR Flow Control

When AAL5-ABR is enabled, the ATM controller implements the ATM Forum TM 4.0 available-bit-rate flow. It automatically inserts forward- and backward-RM cells into the user cell stream and adjusts the transmission rate according to the backwards RM cell feedback; see Section 35.10.2.2.2, “AAL5-ABR Protocol-Specific RCT.” The ABR flow is controlled on a per-VC basis.

35.3 ATM Pace Control (APC) Unit

The ATM pace control (APC) unit schedules the ATM channels for transmitting. While performing this task, the APC unit uses the following parameters:

- Frequency (bandwidth) of each ATM channel
- ATM traffic pacing—Peak cell rate (PCR), sustain cell rate (SCR), and minimum rate (MCR)
- Priority level—Real-time channels (CBR or VBR-RT) are scheduled at high-priority levels; non-real-time channels (VBR-NRT, ABR, UBR) are scheduled at low-priority levels. Up to eight priority levels are available.

35.3.1 APC Modes and ATM Service Types

The ATM Forum (http://www.atmforum.com) defines the service types described in Table 35-1.

Table 35-1. ATM Service Types

Service Type	Cell Rate Pacing	Real-Time/ Non-Real-Time	Relative Priority
CBR	PCR	RT	1 (highest)
VBR-RT	PCR, SCR (peak-and-sustain)	RT	2
VBR-NRT	PCR, SCR (peak-and-sustain)	NRT	3
ABR [1]	PCR	NRT	4
UBR+	PCR, MCR (peak-and-minimum)	NRT	5
UBR	PCR	NRT	6 (lowest)

[1] When ABR flow control is active, the CP automatically adapts the APC parameters PCR, PCR_FRACTION. These parameters function as the channel's allowed cell rate (ACR).

For information about cell rate pacing, see Section 35.3.5, "ATM Traffic Type." For information about prioritization, see Section 35.3.6, "Determining the Priority of an ATM Channel."

35.3.2 APC Unit Scheduling Mechanism

The APC unit consists of an APC data structure in the dual-port RAM for each PHY and a special scheduling algorithm performed by the CP. Each PHY's APC data structure includes three elements: an APC parameter table, an APC priority table, and cell transmission scheduling tables for each priority level. (See Section 35.10.4, "APC Data Structure.")

Each PHY's APC parameter table holds parameters that define the priority table location, the number of priority levels, and other APC parameters. The priority table holds pointers that define the location and size of each priority level's scheduling table.

Each scheduling table is divided into time slots, as shown in Figure 35-1. The user determines the number of ATM cells to be sent each time slot (cells per slot). After a channel is sent, it is removed from the current time slot and advanced to a future time slot according to the channel's assigned traffic rate (specified in time slots). The PCR parameter in the TCT, or the SCR or MCR parameters in the TCT extension (TCTE) determine the channel's actual rate.

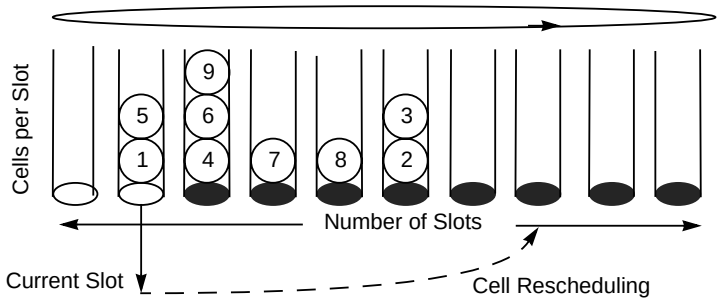


Figure 35-1. APC Scheduling Table Mechanism

Each 2-byte time-slot entry points to one ATM channel. Additional channels scheduled to transmit in the same slot are linked to each other using the APC linked-channel field in the TCT. The linked list is not limited; however, if the number of channels for the current slot exceeds the cells per slot parameter (CPS), the extra channels are sent in subsequent time slots. (The rescheduling of extra channels is based on the original slot to maintain each channel's pace.)

Note that a channel can appear only once in the scheduling table at a given time, because each channel has only one APC linked-channel field.

35.3.3 Determining the Scheduling Table Size

The following sections describe how to determine the number of cells sent per time slot and the total number of slots needed in a scheduling table.

35.3.3.1 Determining the Cells Per Slot (CPS) in a Scheduling Table

The number of cells sent per time slot is determined by the channel with the maximum bit rate; see equation A. The maximum bit rate is achieved when a channel is rescheduled to the next slot. For example, if the line rate is 155.52 Mbps and there are eight cells per slot, equation A yields a maximum VC rate of 19.44 Mbps.

$$\text{Max bit rate} = \frac{\text{line rate}}{\text{cells per slot}} \qquad \text{(A)}$$

Note that a channel can appear only once per time slot; thus, 19.44 Mbps = 155.52Mbps/8.

The cells per slot parameter (CPS) affects the cell delay variation (CDV). Because the APC unit does not put cells in a definite order within each time slot (LIFO—last-in/first-out implementation), as CPS increases, the CDV increases. However as CPS decreases, the size of the scheduling table in the dual-port RAM increases and more CPM bandwidth is required.

35.3.3.2 Determining the Number of Slots in a Scheduling Table

The number of time slots in a scheduling table is determined by the channel with the minimum bit rate; see equation B. The minimum bit rate is achieved when the channel reschedules only once in a whole table scan. (The maximum schedule advance allowed is equal to number_of_slots-1.) For example, if the line rate is 155.52 Mbps, the minimum bit rate is 32 kbps and the CPS is 4, then, according to equation B, the number of slots should be 1,216.

NOTE

The following APC configuration is not recommended for values outside the following ranges (PCR = peak cell rate, PCRF = peak cell rate fraction, NOS = number of slots):

PCR = 1 and PCRF = 0

PCR = NOS - 1 and PCRF = 0

(B) $$\text{Min bit rate} = \frac{\text{line rate}}{(\text{number_of_slots} - 1) \times \text{cells per slot}}$$

For the above example, 32 kbps = 155.52 Mbps/((1216-1) × 4).

Use equations (A) and (B) to obtain the maximum and minimum bit rates of a scheduling table. For example, given a line rate = 155.52 Mbps, number_of_slots = 1025, and CPS = 8:

$$\text{Max bit rate} = (155.52 \text{ Mbps})/8 = 19.44 \text{ Mbps}$$

$$\text{Min bit rate} = (155.52 \text{ Mbps})/(1024 \times 8) = 18.98 \text{ kbps}.$$

35.3.4 Determining the Time-Slot Scheduling Rate of a Channel

The time-slot scheduling rate of each ATM channel is defined by equation C. The resulting number of APC slots is written in either TCT[PCR], TCTE[SCR] or TCTE[MCR], depending on the traffic type.

(C) $$\text{Rate [slots]} = \frac{\text{line rate [bps]}}{\text{VC rate [bps]} \times \text{cells per slot}}$$

35.3.5 ATM Traffic Type

The APC uses the cell rate pacing parameters (PCR, SCR, and MCR) to generate CBR, VBR, ABR, UBR+, and UBR traffic. The user determines the kind of traffic that is generated per VC by writing to TCT[ATT] (ATM traffic type); see Section 35.10.2.3, “Transmit Connection Table (TCT).”

35.3.5.1 Peak Cell Rate Traffic Type

When the peak cell rate traffic type is selected, the APC schedules channels to transmit according to the PCR and PCR_FRACTION traffic parameters. Other traffic parameters do not apply to this traffic type.

35.3.5.2 Determining the PCR Traffic Type Parameters

Suppose a VC uses 15.66 Mbps of the total 155.52 Mbps and CPS = 8. Equation C yields:

$$\text{PCR [slots]} = (155.52 \text{ Mbps})/(15.66 \text{ Mbps} \times 8) = 1.241$$

The resulting number of slots is written into TCT[PCR] and TCT[PCR_FRACTION]. Because PCR_FRACTION is in units of 1/256 slots, the fraction must be converted as follows:

$$1.241 = 1 + 0.241 \times 256/256 = 1 + 61.79/256 \sim 1 + 62 / 256$$

PCR = 1 PCR_FRACTION = 62

35.3.5.3 Peak and Sustain Traffic Type (VBR)

Variable bit rate (VBR) traffic can burst at the peak cell rate as long as the long-term average rate does not exceed the sustainable cell rate. To support VBR channels, the APC implements the GCRA (generic cell rate algorithm) using three parameters—the peak cell rate (PCR), the sustained cell rate (SCR), and burst tolerance (BT), as shown in Figure 35-2. (The GCRA is also known as the leaky bucket algorithm.)

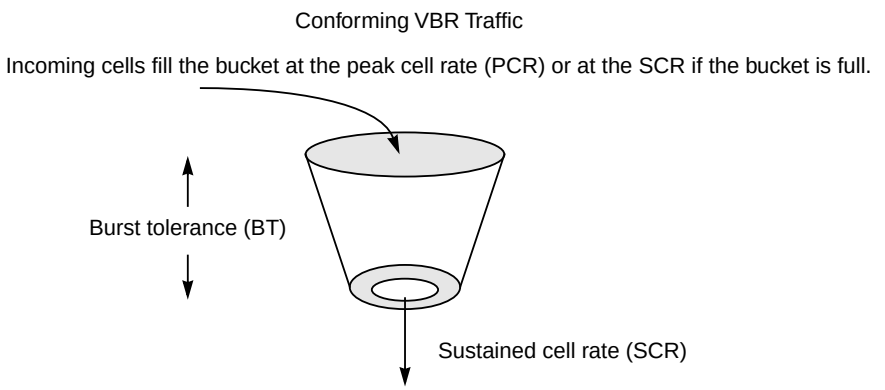


Figure 35-2. VBR Pacing Using the GCRA (Leaky Bucket Algorithm)

When a VBR channel is activated, it bursts at the peak cell rate (PCR) until reaching its initial burst tolerance (BT), which is the buffer length the network allocated for this VC. When the burst limit is reached, the APC reduces the VC's scheduling rate to the sustained cell rate (SCR). The VC continues sending at SCR as long as TxBDs are ready. However, as each SCR time allotment elapses with no TxBD ready to send, the APC grants the VC a

credit for bursting at the peak cell rate (PCR). (Gaining credit implies that the buffer at the switch is not full and can tolerate a burst transmission.) If a TxBD becomes ready, the APC schedules the VC to burst at the PCR as long as credit remains. When the burst credit ends (the network's UPC leaky bucket reaches its limit), the APC schedules the VC according to SCR.

35.3.5.3.1 Example for Using VBR Traffic Parameters

Suppose the traffic parameters of a VBR channel are PCR = 6 Mbps, SCR = 2 Mbps, MBS (maximum burst size) = 1000 cells, and CPS = 8.

Equation C (see Section 35.3.4, "Determining the Time-Slot Scheduling Rate of a Channel") yields the APC parameters, PCR, PCR_FRACTION, SCR, and SCR_FRACTION, which the user writes to the channel's TCT.

PCR [slots] = (155.52 Mbps)/(6 Mbps × 8) = 3.24

3.24 = 3 + 0.24 × 256/256 = 3 + 61.44/256 ~ 3 + 62/256

PCR = 3 PCR_FRACTION = 62

SCR [slots] = (155.52 Mbps)/(2 Mbps × 8) = 9.72

9.72 = 9 + (0.72 × 256/256) = 9 + 184.32/256 ~ 9 + 185/256

SCR = 9 SCR_FRACTION = 185

Equation D yields the number of slots the user writes to the channel's TCT[BT].

(D)

$$\begin{aligned} BT\,[slots] &= (MBS[cells] - 2) \times (SCR[slots] - PCR[slots]) + SCR[slots] \\ &= (1000 - 2) \times ((9+185/256) - (3+62/256)) + (9 +185/256) \\ &= 6477 \end{aligned}$$

35.3.5.3.2 Handling the Cell Loss Priority (CLP)—VBR Type 1 and 2

The MPC8560 supports two ways to schedule VBR traffic based on the cell loss priority (CLP). When TCTE[VBR2] is cleared, CLP_{0+1} cells are scheduled by PCR or SCR according to the GCRA state. When TCTE[VBR2] is set, CLP_0 cells are still scheduled by PCR or SCR according to the GCRA state, but CLP_1 cells are always scheduled by PCR. See Section 35.10.2.3.4, "VBR Protocol-Specific TCTE."

35.3.5.4 Peak and Minimum Cell Rate Traffic Type (UBR+)

To support UBR+ channels, the APC schedules transmission according to PCR and MCR. For each priority level, the APC maintains a parameter that monitors the traffic load measured as the time-slot delay between the service pointer (pointing to the current time slot waiting transmission) and a real-time slot pointer. If the transmission delay is greater

than MDA (maximum delay allowed), the APC begins scheduling channels according to the MCR parameter. If the delay, however, drops below MDA, the APC again schedules channels according to the PCR. Note that in order to guarantee a minimum cell rate for UBR+ channels, there must be enough bandwidth to simultaneously send all possible channels at the MCR. See Section 35.10.2.3.5, “UBR+ Protocol-Specific TCTE.”

35.3.6 Determining the Priority of an ATM Channel

The priority mechanism is implemented by adding priority table levels, which point to separate scheduling tables; see Section 35.10.4, “APC Data Structure.” The APC flow control services the APC_LEVEL1 slots first. If there are no cells to send, the APC goes to the next priority level. The APC has up to eight priority levels with APC_LEVEL8 being the lowest. The user specifies the priority of an ATM channel when issuing the ATM TRANSMIT command; see Section 35.14, “ATM Transmit Command.”

The real-time channels, CBR and VBR-RT, should be inserted in APC_LEVEL1; non-real-time channels, VBR-NRT, ABR, and UBR should be inserted in lower priority levels.

35.4 VCI/VPI Address Lookup Mechanism

The MPC8560 supports two ways to look up addresses for incoming cells:

- External CAM lookup
- Address compression

Writing to GMODE[ALM] (address-lookup-mechanism bit) in the parameter RAM selects the mechanism. Both mechanisms are described in the following sections.

35.4.1 External CAM Lookup

An external CAM is usually used when the range of VCI/VPI values varies widely or is unknown. Clearing GMODE[ALM] selects the external CAM address lookup mechanism. If there is no match in the external CAM, the cell is considered a misinserted cell. The external CAM can point to internal or external channels (channels whose connection table resides in external memory). The CAM input, shown in Figure 35-3, is the 32-bit cell address: PHY address, GFC + VPI, and VCI.

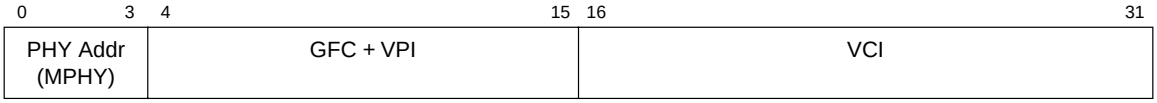


Figure 35-3. External CAM Data Input Fields

The output of the CAM, shown in Figure 35-4, is a 32-bit entry (16-bit channel code and a match-status bit).

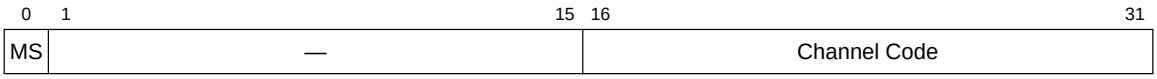


Figure 35-4. External CAM Output Fields

The external CAM fields are described in Table 35-2.

Table 35-2. External CAM Input and Output Field Descriptions

Field	Description
PHY Addr	In multiple PHY mode, this field contains the 4 least-significant bits of the current channel's physical address. Because this CAM comparison field is limited to 4 bits, two CAM devices are needed if using more than 16 PHYs.The msb of the PHY address (bit 4) selects between the two devices. If the msb is zero, the CP accesses the CAM whose address is written in the EXT_CAM_BASE parameter in the parameter RAM; if the msb is set, the CP uses EXT_CAM1_BASE. See Section 23.4.1, "CMX UTOPIA Address Register (CMXUAR)." In single PHY mode, clear this field.
GFC+VPI, VCI	The GFC, VPI, and VCI of the current channel.
Ch Code	Pointer to internal or external connection table.
—	Reserved, should be cleared.
MS	Match status. 0 Match was found. 1 Match was not found.

35.4.2 Address Compression

The address compression mechanism uses two levels of address translation to help minimize the memory space needed to cover the available address range. The first level of translation (VP-level) uses a look-up table based on the 4-bit PHY address and the 12-bit virtual path identifier; the second level (VC-level) uses the 16-bit virtual channel identifier. If there is no match during address compression, the cell is considered a misinserted cell.

During the VP-level translation, VP_MASK in the ATM parameter RAM compresses an incoming cell's PHY address and VPI to create an index into the VP-level table. The VP-level table entry consists of another mask (VC_MASK) and a pointer to one of the VC-level tables (VCOFFSET). Note that the VP table should reside in the dual-port RAM.

In the VC-level translation, the VCI is compressed with the VC_MASK to generate a pointer to the VC-level table entry containing the received cell's channel code. The VC table should reside in external memory.

Figure 35-5 shows an example of address compression.

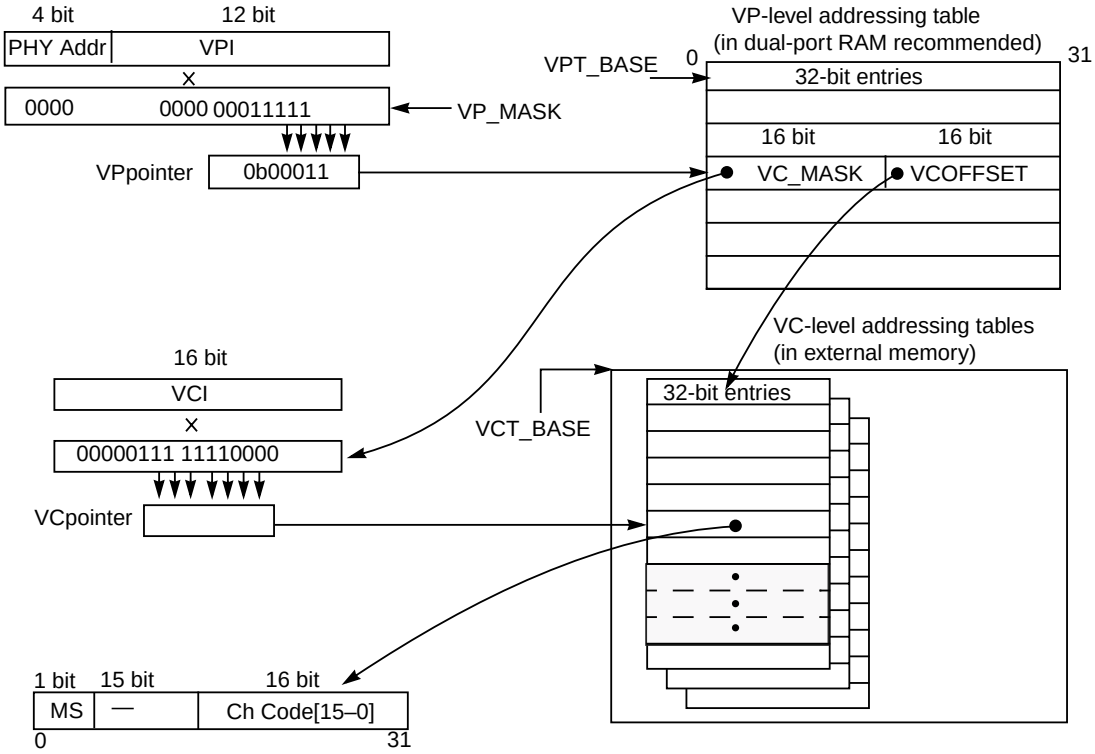


Figure 35-5. Address Compression Mechanism

Figure 35-5 shows VP_MASK selecting five VPI bits to index the VP-level table. The VP-level table entry contains the 16-bit mask (VC_MASK) and the VC-level table offset (VCOFFSET) for the next level of address mapping. The VC_MASK selects VCI bits 4–10, which is used with VCT_BASE and VCOFFSET to indicate the received cell's channel code.

Table 35-3 describes the address compression fields.

Table 35-3. Field Descriptions for Address Compression

Field	Description
PHY Addr	In multiple PHY mode, this field contains the 4 least-significant bits of the current channel's physical address. Because this comparison field is limited to 4 bits, two sets of look-up tables are needed if using more than 16 PHYs.The msb of the PHY address (bit 4) selects between the two sets of tables. If the msb is zero, the CP accesses the tables at VPT_BASE and VCT_BASE; if the msb is set, the CP uses VPT1_BASE and VCT1_BASE. See Section 23.4.1, "CMX UTOPIA Address Register (CMXUAR)." In single PHY mode, clear this field.
VCI, VPI	The VCI and VPI of the current channel.
Ch Code	Pointer to internal or external connection table.

Table 35-3. Field Descriptions for Address Compression

Field	Description
—	Reserved, should be cleared.
MS	Match status. 0 Match was found. 1 Match was not found.

35.4.2.1 VP-Level Address Compression Table (VPLT)

The size of the VP-level table depends on the number of mask bits in VP_MASK. For example, if only one PHY is available (PHY address = 0) and VPMASK = 0b11_1111_1111, VP pointer contains ten bits and the table is 4 Kbytes. Because each VPLT entry is 4 bytes, the address of an entry is VPT_BASE + VP pointer × 4.

Each VPLT entry has two parameters:

- VC_MASK—A 16-bit VC-level mask for masking the incoming cell's VCI
- VCOFFSET—A 16-bit VC-level table offset from VC_BASE that points to the appropriate VC-level table's (VCLT) starting address. The address of the VCLT is VC_BASE + VCOFFSET × 4.

  If the VCLTs are to be placed contiguously in memory, each table's VCOFFSET depends on the size of preceding tables. Each table's size depends on the number of ones in VC_MASK. Figure 35-6 gives the general formula for determining VCOFFSET.

General formula: $VCOFFSET_{(n+1)} = VCOFFSET_n + 2^{(\text{number of ones in VC_MASKn})}$

Figure 35-6. General VCOFFSET Formula for Contiguous VCLTs

Table 35-4 shows example VCOFFSET calculations for a VP-level table with four entries.

Table 35-4. VCOFFSET Calculation Examples for Contiguous VCLTs

VP-Level Table Entry	VC_MASK	Number of Ones in VC_MASK	VC-Level Table Size	VCOFFSET
0	0x0237	6	2^6 = 64 entries	0
1	0x0230	3	2^3 = 8 entries	64
2	0xA007	5	2^5 = 32 entries	64 + 8 = 72
3	x	x	x	72 + 32 = 104

The MPC8560 can check that all unallocated bits of the PHY + VPI are 0 by setting GMODE[CUAB] (check unallocated bits) in the parameter RAM. If they are not, the cell is considered a misinserted cell.

Table 35-5 gives an example of VP-level table entry address calculation.

Table 35-5. VP-Level Table Entry Address Calculation Example

VPT_BASE	VP-Level Table Size	VP_MASK	Phy+VPI	VP Pointer	VP Entry Address
0x0024_0000	64 entries	0x0237	0x0011	0x09	VP Base = 0x240000 0x09 x 4 = 0x000024 0x240024

Figure 35-7 shows the VP pointer address compression from Table 35-5.

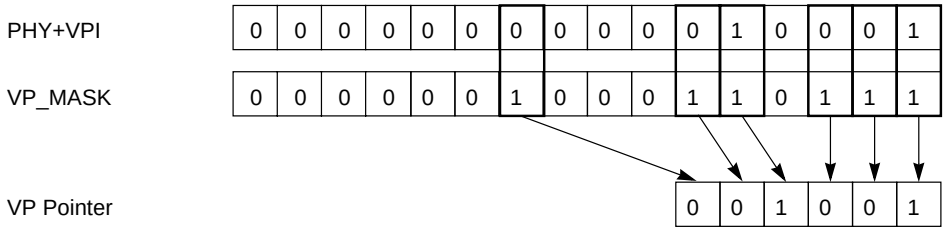


Figure 35-7. VP Pointer Address Compression

35.4.2.2 VC-Level Address Compression Tables (VCLTs)

Each VPLT entry points to a single VCLT. Like the VPLT, the size of each VCLT depends on VC_MASK. Because the VCLT contains word entries, if VC_MASK = 0b11_1111_1111, the table is 4 Kbytes. The address of an entry in this table is VCT_BASE + VCOFFSET × 4 + VCpointer × 4.

The MPC8560 can check that all unallocated VCI bits are 0 by setting GMODE[CUAB] (check unallocated bits). If they are not, the cell is considered a misinserted cell.

An example of VC-level table entry address calculation is shown in Table 35-6. Note that VCOFFSET is assumed to be 0x100 for this example.

Table 35-6. VC-Level Table Entry Address Calculation Example

VCT_BASE	VCOffset	VC-Level Table Size	VC_MASK	VCI	VC Pointer	VC Entry Address
0x0084_0000	0x0100	32 entries	0x0037	0x0031	0x19	VC Base = 0x840000 0x100 x 4 = 0x000400 0x19 x 4 = 0x000064 0x840464

Figure 35-8 shows the VC pointer address compression from Table 35-6.

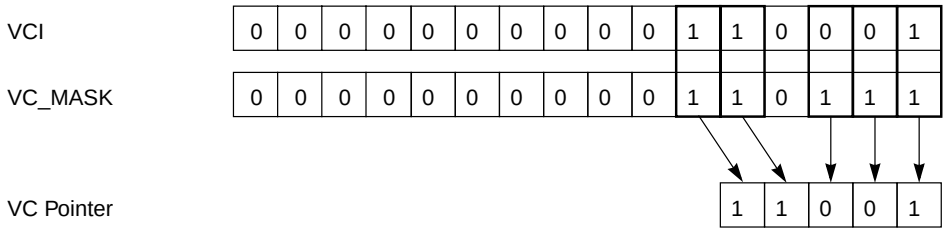


Figure 35-8. VC Pointer Address Compression

35.4.3 Misinserted Cells

If the address lookup mechanism cannot find a match (MS=1), the cell is discarded and ATM layer statistics are updated, as described in Section 35.8, “ATM Layer Statistics.”

35.4.4 Receive Raw Cell Queue

Channel one in the RCT is reserved as a raw cell queue. The user should program channel one to operate in AAL0 protocol. The receive raw cell queue is used for removing management cells from the regular cell flow to the host. When a management cell is sent to the receive raw cell queue, the CP sets RxBD[OAM]. The ALL0 BD specifies the channel code associated with the current OAM cell.

The following are optionally removed from the regular flow and sent to the raw cell queue:

- Segment F5 OAM (PTI = 0b100). To enable F5 segment filtering, set RCT[SEGF].
- End-to-end F5 OAM (PTI = 0b101). To enable F5 end-to-end filtering, set RCT[ENDF].
- RM cells (PTI = 0b110). When ABR flow is enabled the cells are terminated internally; otherwise, they are sent to the raw cell queue.
- Reserved PTI value (PTI = 0b111). Always sent to the raw cell queue.
- VCI value: 3, 4, 6, 7–15. To enable VCI filtering set the associated bit in the VCIF entry in the parameter RAM.

Figure 35-9 shows a flowchart of the ATM cell flow.

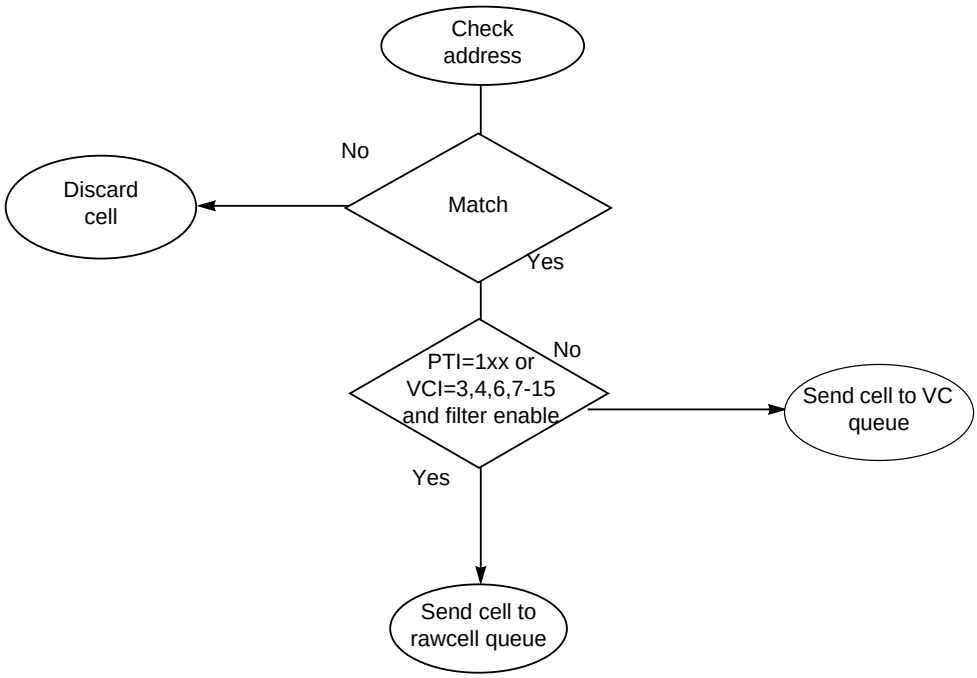


Figure 35-9. ATM Address Recognition Flowchart

Note that even reserved VCI channels should appear in the CAM or address compression tables; otherwise, a cell on a reserved channel will be considered misinserted.

35.5 Available Bit Rate (ABR) Flow Control

While CBR service provides a fixed bandwidth and is useful for real-time applications with strictly bounded end-to-end cell transfer delay and cell-delay variation, ABR service is intended for data applications that can adapt to time-varying bandwidth and can tolerate significant cell transfer delay and cell delay variation. The MPC8560 implements the two following mechanisms defined by the ATM Forum TM 4.0 rate-based flow control.

- Explicit forward congestion indication (EFCI). The network supplies binary indication of whether congestion occurred along the connection path. This information is carried in the PTI field of the ATM cell header (similar to that used in frame relay). The source initially clears each ATM cell's EFCI bit, but as the cell passes through the connection, any congested node can set it. The MPC8560 detects this indication and sets the congestion indication (CI) bit in the next backwards RM cell to signal the source end station to reduce its transmission rate.
- Explicit rate (ER) feedback. The network carries explicit bandwidth information, to allow the source to adjust its rate. The source sends forward RM cells specifying its chosen transmit rate (source ER). A congested switch along the network may

decrease ER to the exact rate it can support. The destination receives forward RM cells and returns them to the source as backward RM cells. The MPC8560 implements source behavior by adjusting the rate according to each returning backward RM cell's ER.

Explicit rate feedback has several advantages over binary feedback (EFCI). Explicit rate feedback allows immediate source rate adaptation, eliminating rate oscillation caused by incremental rate changes. Using the information in RM cells, the network can allocate bandwidth evenly among active ABR channels.

35.5.1 The ABR Model

Figure 35-10 shows the MPC8560 ABR model.

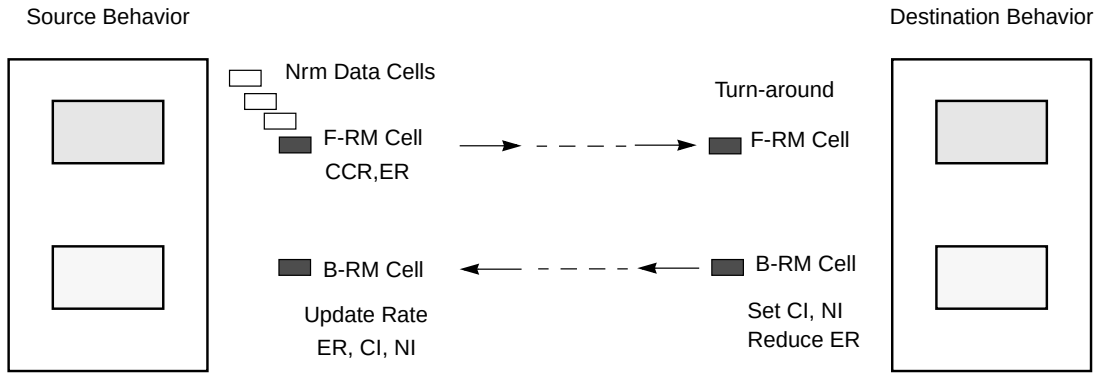


Figure 35-10. MPC8560 ABR Basic Model

The MPC8560 ABR flow control implements both source and destination behavior. The MPC8560 ABR flowchart is described in Section 35.5.1.3, "ABR Flowcharts."

35.5.1.1 ABR Flow Control Source End-System Behavior

The MPC8560's implementation of ABR flow control for end-system sources is described in the following steps:

1. An ABR channel's allowed cell rate (ACR) lies between the minimum cell rate (MCR) and the peak cell rate (PCR).
2. ACR is initialized to the initial cell rate (ICR).
3. An F-RM (Forward-RM) cell is sent for every Nrm data cell sent. If more than Mrm cells are sent and the time elapsed since the last F-RM exceeds Trm, an F-RM cell is sent.
4. When sending an F-RM cell, the current ACR is written in the CCR (current cell rate) field of the RM cell.

5. When B-RM (backward-RM) cell is received with CI = 1 (congestion indication), ACR is reduced by ACR × RDF (rate decrease factor). After the reduction, the new ACR is determined first by letting ACRtemp be the min of (ACR, ER), and then taking the max of (ACRtemp, MCR).
6. When B-RM is received with CI=0 and NI=0 (no increase), ACR is increased by RIF × PCR (rate increase factor). The new ACR is determined first by letting ACRtemp be the min of (ACR, ER), and then taking the max of (ACRtemp, MCR).
7. Before sending an F-RM cell, if more than ADTF (ACR decrease time factor) has elapsed since sending the last F-RM cell, ACR is reduced to ICR. In other words, if the source does not fully use its gained bandwidth, it loses it and resumes sending at its initial cell rate.
8. Before sending an F-RM cell and after action 7, if more than Crm F-RM cells were sent since the last B-RM cell was received with BN=0 (backward notification), the ACR is reduced by ACR × CDF (cutoff decrease factor).
9. A source whose ACR is less than the tag cell rate (TCR) sends out-of-rate cells at the TCR. This behavior is intended for sources whose rates were set to zero by the network. These sources should periodically sense the network state by sending out-of-rate RM cells. In this case data cells will not be sent.
10. An RM cell with an incorrect CRC10 is discarded and the UNI statistics tables are updated.

35.5.1.2 ABR Flow Control Destination End-System Behavior

The MPC8560's implementation of ABR flow control for end-system destinations is described in the following steps:

1. A received F-RM cell is turned around and sent as a B-RM cells.
2. The DIR field of the received F-RM cell is changed from 0 to 1 (backward DIR).
3. The CCR and MCR fields are taken from the F-RM and is not changed.
4. The CI bit of the B-RM cell is set if the previous data cell arrived with EFCI = 1 (congestion bit in the ATM cell header).
5. The ER field of the turn around B-RM cells is limited by TCTE[ER-BRM].
6. If a F-RM cell arrives before the previous F-RM cell was turned around (for the same connection), the new RM cell overwrites the old RM cell.

35.5.1.3 ABR Flowcharts

The MPC8560's ABR transmit and receive flow control is described in the following flowcharts. See Figure 35-11, Figure 35-12, Figure 35-13, and Figure 35-14.

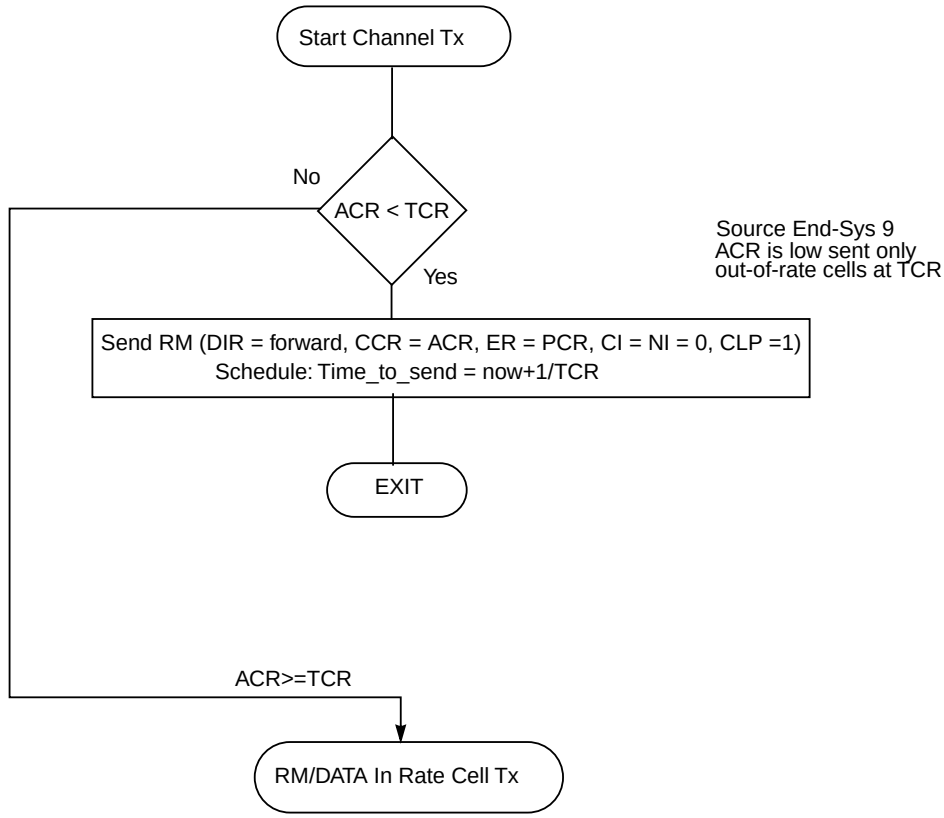


Figure 35-11. ABR Transmit Flow

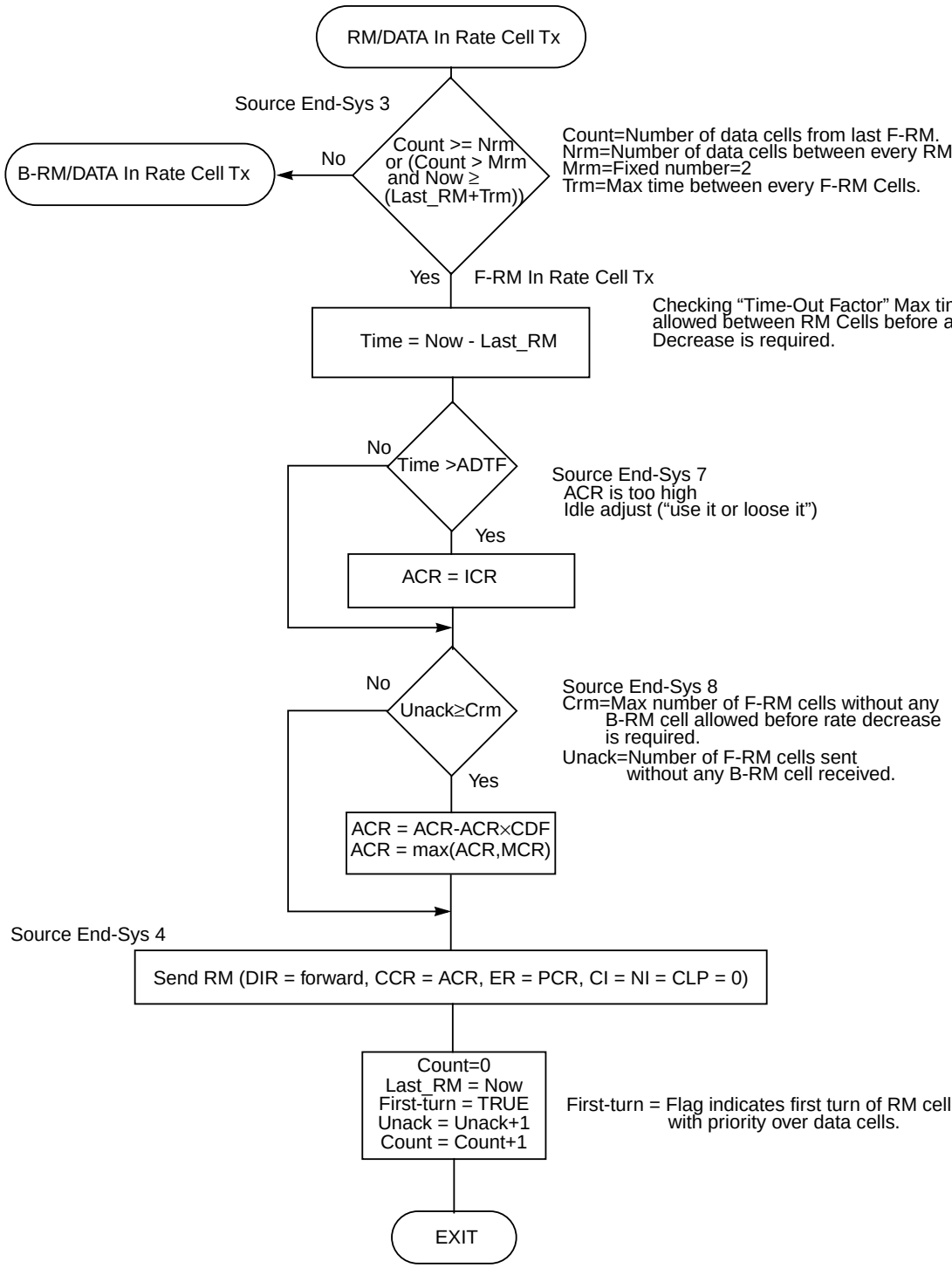


Figure 35-12. ABR Transmit Flow (Continued)

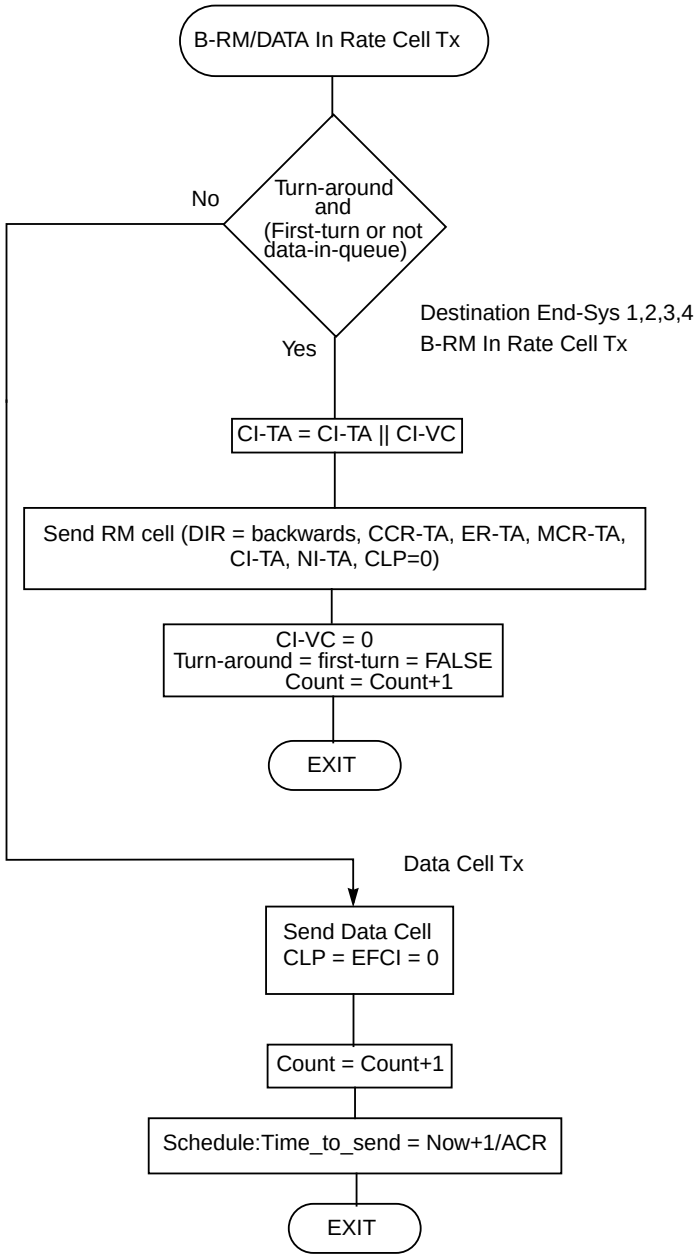


Figure 35-13. ABR Transmit Flow (Continued)

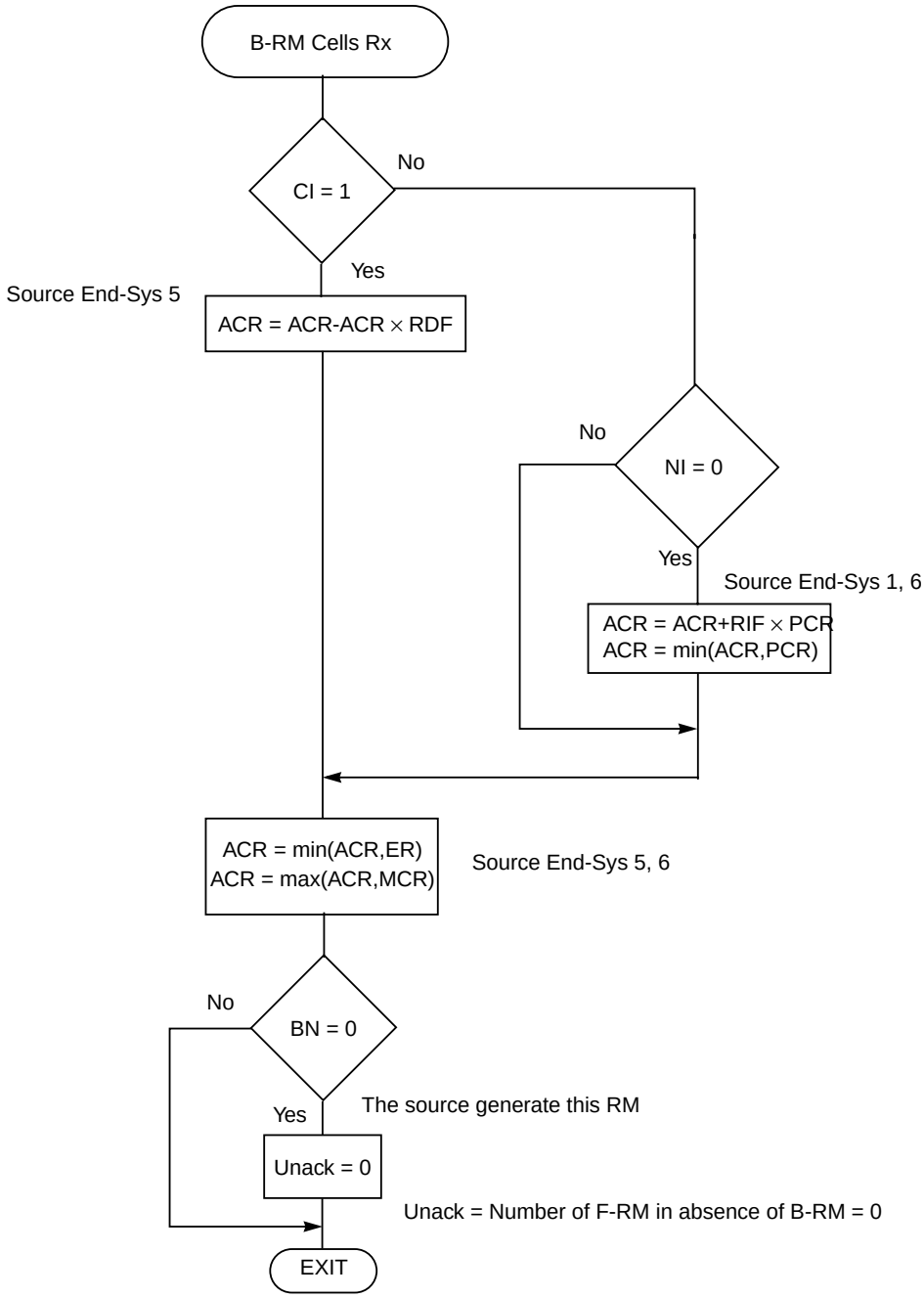


Figure 35-14. ABR Receive Flow

35.5.2 RM Cell Structure

Table 35-7 describes the structure of the RM cell supported by the MPC8560. For more information, see the ABR flow-control traffic management specification (TM 4.0) on the ATM Forum website at http://www.atmforum.com.

Table 35-7. Fields and their Positions in RM Cells

Fields	Octet	Bits	Description	Value
Header	1–5	All	ATM cell header	RM-VCC PTI=6
ID	6	All	Protocol ID	1
DIR	7	0	Direction of RM cell (0 = forward, 1 = backward)	
BN	7	1	Backward notification (BN = 0, the cell was generated by the source; BN=1, the cell was generated by the network or by the destination)	
CI	7	2	Congestion indication. (1 = congestion, 0 = otherwise)	
NI	7	3	No increase indication. (1 = no increase allowed, 0 = otherwise)	
RA	7	4	Not used (ATM Forum ABR)	0
—	7	5-7	Reserved, should be cleared.	0
ER	8–9	All	Explicit rate; see Section 35.5.2.1	
CCR	10–11	All	Current cell rate; see Section 35.5.2.1	
MCR	12–13	All	Minimum cell rate; see Section 35.5.2.1	
QL	14–17	All	Not used (ATM Forum ABR)	0
SN	18–21	All	Not used (ATM Forum ABR)	0
—	22–51	All	Reserved, should be cleared.	0x6A for each byte
—	52	0–5	Reserved, should be cleared.	0
CRC-10	52	6–7	CRC-10	
	53	All		

35.5.2.1 RM Cell Rate Representation

Rates in the RM cells are represented in a binary floating-point format using a 5-bit exponent (e), a 9-bit mantissa (m), and a 1-bit nonzero flag (nz), as shown in Figure 35-15.

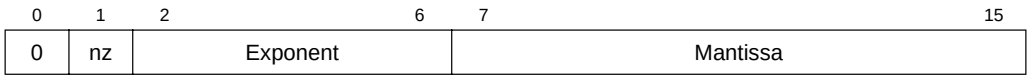


Figure 35-15. Rate Format for RM Cell

The rate (in cells/second) is calculated as in Figure 35-16.

$$\text{Rate} = \left[2^{e} \times \left(1 + \frac{m}{512}\right)\right] \times nz$$

Figure 35-16. Rate Formula for RM Cells

Initialize the traffic parameters (ER, MCR, PCR, or ICR) in the ABR protocol-specific connection tables using the rate formula in Figure 35-16.

35.5.3 ABR Flow Control Setup

Follow these steps to setup ABR flow control:

1. Initialize the ABR data structure: RCT, TCT, RCT-ABR protocol-specific, TCTE-ABR protocol-specific.
2. Initialize ABR global parameters in the parameter RAM. See Section 35.10.1, “Parameter RAM.”
3. Program the AAL-type in the RCT and TCT to AAL5 and set TCT[ABRF]. Note that the ABR flow control is available only with AAL5.
4. The time stamp timer generates the RM cell’s time stamp, which the ABR flow control monitors to maintain source behavior in steps #3 and #7 of Section 35.5.1.1, “ABR Flow Control Source End-System Behavior.” Enable the time stamp timer by writing to the RTSCR; see Section 20.3.7, “RISC Time-Stamp Control Register (RTSCR).”
5. Initialize the ABR parameters (CPS_ABR and LINE_RATE_ABR) in the APCT; see Section 35.10.4.1, “APC Parameter Tables.” Note that when using ABR, the CPS (cells per slot) parameter in the APCPT should be a power of two.
6. Finally, send the ATM TRANSMIT command to restart channel transmission.

35.6 OAM Support

This section describes the MPC8560’s support for ATM-layer (F4 out-of-band, and F5 in-band) operations and maintenance (OAM) of connections. Alarm surveillance, continuity checking, remote defect indication, and loopback cells are supported using OAM receive and transmit AAL0 cell queues. Using dedicated support, performance management block tests can be performed on up to 64 connections simultaneously. The CP automatically inserts forward monitoring cells (FMC) and generates backward-reporting cells (BRC) as recommended by ITU I.610.

35.6.1 ATM-Layer OAM Definitions

Table 35-8 lists pre-assigned header values at the user-network interface (UNI).

Table 35-8. Pre-Assigned Header Values at the UNI

Use	GFC	VPI	VCI	PTI	CLP
Segment OAM F4 flow cell	xxxx	aaaa_aaaa	0000_0000_0000_0011	0a0	a
End-to-end OAM F4 flow cell	xxxx	aaaa_aaaa	0000_0000_0000_0100	0a0	a
Segment OAM F5 flow cell	xxxx	aaaa_aaaa	aaaa_aaaa_aaaa_aaaa	100	a
End-to-end OAM F5 flow cell	xxxx	aaaa_aaaa	aaaa_aaaa_aaaa_aaaa	101	a

a = available for use by the appropriate ATM layer function

Table 35-9 lists pre-assigned header values at the network-node interface (NNI).

Table 35-9. Pre-Assigned Header Values at the NNI

Use	VPI	VCI	PTI	CLP
Segment OAM F4 flow cell	aaaa_aaaa_aaaa	0000_0000_0000_0011	0a0	a
End-to-end OAM F4 flow cell	aaaa_aaaa_aaaa	0000_0000_0000_0100	0a0	a
Segment OAM F5 flow cell	aaaa_aaaa_aaaa	aaaa_aaaa_aaaa_aaaa	100	a
End-to-end OAM F5 flow cell	aaaa_aaaa_aaaa	aaaa_aaaa_aaaa_aaaa	101	a

a= available for use by the appropriate ATM layer function

35.6.2 Virtual Path (F4) Flow Mechanism

The F4 flow is designated by pre-assigned virtual channel identifiers within the virtual path. The following two kinds of F4 flows can exist simultaneously:

- End-to-end (identified as VCI 4)—This flow is used for end-to-end VPC operations communications. Cells inserted into this flow can be removed only by the endpoints of the virtual path.
- Segment (identified as VCI 3)—This flow is used for communicating operations information within one VPC link or among multiple interconnected VPC links. The concatenation of VPC links is called a VPC segment. Cells inserted into this flow can be removed only by the segment endpoints, which must remove these cells to prevent confusion in adjacent segments.

35.6.3 Virtual Channel (F5) Flow Mechanism

The F5 flow is designated by pre-assigned payload type identifiers. The following two kinds of F5 flow can exist simultaneously:

- End-to-end (identified by PTI = 5)—This flow is used for end-to-end VCC operations communications. Cells inserted into this flow can be removed only by VC endpoints.

- Segment (identified by PTI = 4)—This flow is used for communicating operations information with the bound of one VCC link or multiple interconnected VCC links. A concatenation of VCC links is called a VCC segment. Segment endpoints must remove these cells to prevent confusion in adjacent segments.

35.6.4 Receiving OAM F4 or F5 Cells

OAM F4/F5 flow cells are received using the raw cell queue, described in Section 35.4.4, “Receive Raw Cell Queue.” An F4/F5 OAM cell which does not appear in the CAM or address compression tables is considered a misinserted cell.

35.6.5 Transmitting OAM F4 or F5 Cells

OAM F4/F5 flow cells are sent using the usual AAL0 transmit flow. For OAM F4/F5 cell transmission, program channel one in the TCT to operate in AAL0 mode. Enable the CR10 (CRC-10 insertion) mode as described in Section 35.10.2.3.3, “AAL0 Protocol-Specific TCT.” Prepare the OAM F4/F5 flow cell and insert it in an AAL0 TxBD. Finally, issue a ATM TRANSMIT command to send the OAM cell. For multiple PHYs, use several AAL0 channels—each PHY should have one transmit raw cell queue that is associated with its scheduling table.

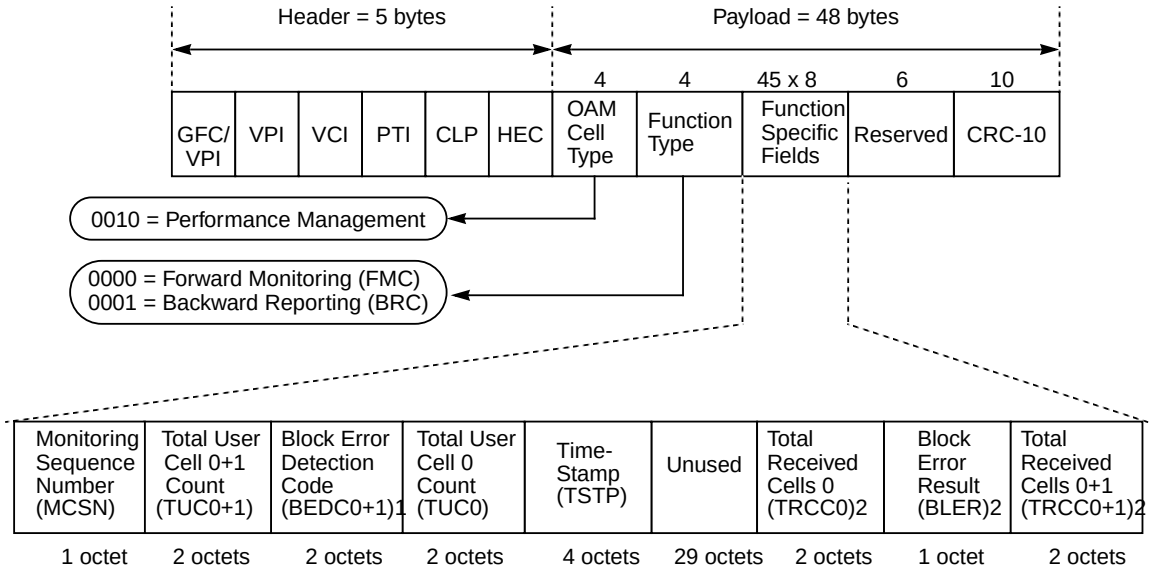
A series of OAM cells can be sent using one ATM TRANSMIT command by creating a table of AAL0 TxBDs. If the channel’s TCT[AVCF] (auto VC off) is set, the transmitter automatically removes it from the APC (that is, it does not generate periodic transmit requests for this channel after all AAL0 BDs are processed).

35.6.6 Performance Monitoring

A connection’s performance is monitored by inspecting blocks of cells (delimited by forward monitoring cells) sent between connection or segment endpoints. Each FMC contains statistics about the immediately preceding block of cells. When an endpoint receives an FMC, it adds the statistics generated locally across the same block to produce a backward reporting cell (BRC), which is then returned to the opposite endpoint.

The MPC8560 can run up to 64 bidirectional block tests simultaneously. When a bidirectional test is run, FMCs are generated for one direction and checked for the opposite.

Figure 35-17 shows the FMC and BRC cell structure.



- 1. $BEDC_{0+1}$ appears in FMCs only.
- 2. $TRCC_0$, BLER, and $TRCC_{0+1}$ appear in BRCs only.

Figure 35-17. Performance Monitoring Cell Structure (FMCs and BRCs)

Table 35-10 describes performance monitoring cell fields.

Table 35-10. Performance Monitoring Cell Fields

Field	Description	BRC	FMC
MCSN	Monitoring cell sequence number. The sequence number of the performance monitoring cell (modulo 256).	Yes	Yes
TUC_{0+1}	Total user cell 0+1 count. Counts all user cells (modulo 65,536) sent before the FMC was inserted.	Yes	Yes
TUC_0	Total user cell 0 count. Counts CLP = 0 user cells (modulo 65,536) sent before the FMC was inserted.	Yes	Yes
TSTP	Time stamp. Used to indicate when the cell was inserted.	Yes	Yes
$BEDC_{0+1}$	Block error detection code. Even parity over the payload of the block of user cells sent since the last FMC.	No	Yes
$TRCC_0$	Total received cell count 0. Counts CLP=0 user cells (modulo 65,536) received before the FMC was received.	Yes	No
BLER	Block error result. Counts error parity bits detected by the BEDC of the received FMC.	Yes	No
$TRCC_{0+1}$	Total received cell count 0+1. Counts all user cells (modulo 65,536) received before the FMC was received.	Yes	No

35.6.6.1 Running a Performance Block Test

For bidirectional PM block tests, FMCs are monitored at the receive side and generated at the transmit side. The following setup is required to run a bidirectional PM block test on an active VCC:

1. Assign one of the available 64 performance monitoring tables by writing to both RCT[PMT] and TCT[PMT] and initializing the one chosen. See Section 35.10.3, “OAM Performance Monitoring Tables.”
2. For PM F5 segment termination set RCT[SEGF]; for PM F5 end-to-end termination set RCT[ENDF].
3. Finally, set the channel’s RCT[PM] and TCT[PM] and the receive raw cell’s RCT[PM].

For unidirectional PM block tests:

- For PM block monitoring only, set only the RCT fields above.
- For PM block generation only, set only the TCT fields above.

To run a block test on a VPC, assign all the VCCs of the tested VPC to the same performance monitoring table. Configure RCT[PMT] and TCT[PMT] to specify the performance monitoring table associated with each F4 channel.

35.6.6.2 PM Block Monitoring

PM block monitoring is done by the receiver. After initialization (see Section 35.6.6.1), whenever a cell is received for a VCC or VPC, the TRCC counters are incremented and the BEDC is calculated. When an FMC is received, the CP adds the BRC fields into the cell payload ($TRCC_0$, $TRCC_{0+1}$, BLER) and transfers the cell to the receive raw cell queue. The user can monitor the BRC cell results and transfer the cell to the transmit raw cell queue.

Before the BRC is transferred to the transmit raw cell queue, the PM function type should be changed to backward reporting and additional checking should be done regarding the BLER field. If the sequence numbers (MCSN) of the last two FMCs are not sequential or the differences between the last two TUCs and the last two TRCCs are not equal, BLER should be set to all ones (see the ITU I.610 recommendation).

Note that the TRCCs are free-running counters (modulo 65,536) that count user cells received. The total received cells of a particular block is the difference between TRCC values of two consecutive BRC cells. TRCC values are taken from a VC’s performance monitoring table.

35.6.6.3 PM Block Generation

The transmitter generates the PM block. Each time the transmitted cell count parameter (TCC) in the performance monitoring table reaches zero, the CP inserts an FMC into the

user cell stream. The CP copies the FMC header, SN-FMC, TUC_{0+1}, TUC_0, $BEDC_{0+1}$-Tx from the performance monitoring table and inserts them into the FMC payload. The TSTP value (FMC time stamp field) is taken from the MPC8560 time stamp timer; see Section 20.3.7, “RISC Time-Stamp Control Register (RTSCR).”

The TUCs are free-running counters (modulo 65,536) that count transmitted user cells. The total transmitted cells of a particular block is the difference between TUC values of two consecutive FMCs. The BEDC (BIP-16, bit interleaved parity) calculation is done on the payload of all user cells of the current tested block. The performance monitoring block can range from 1 to 2K cells, as specified in the BLCKSIZE parameter in the performance monitoring table; see Section 35.10.3, “OAM Performance Monitoring Tables.”

In Figure 35-18, the performance monitoring block size is 512 cells. For every 512 user cells sent, the ATM controller automatically inserts an FMC into the regular cell stream as defined in ITU I.610. When an FMC is received, the ATM controller adds the BRC fields to the cell payload and sends the cell to the raw cell queue. The user can monitor the BRC cell results and transfer the cell to the transmit raw cell queue.

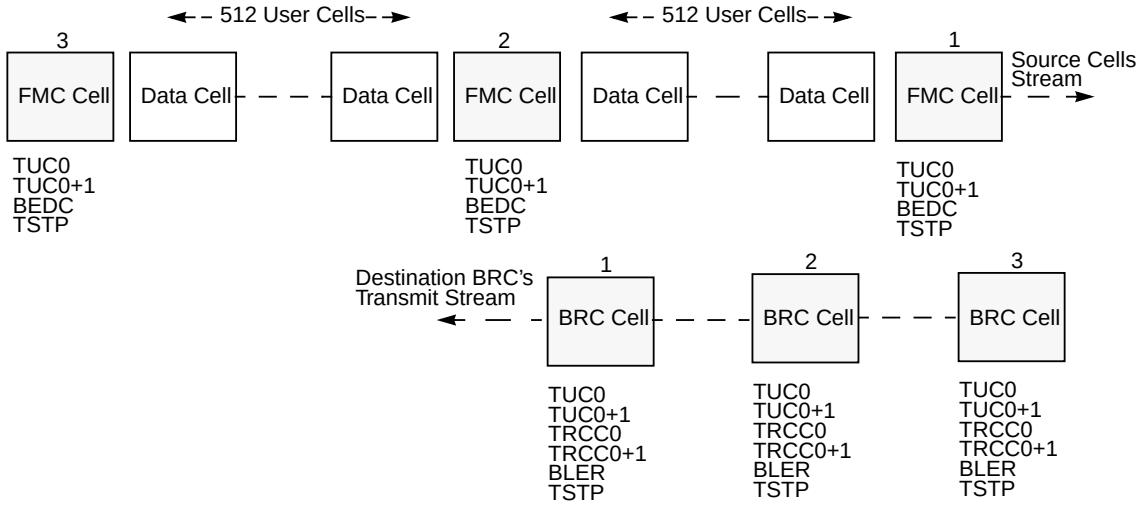


Figure 35-18. FMC, BRC Insertion

35.6.6.4 BRC Performance Calculations

BRC reception uses the regular AAL0 raw cell queue. On receiving two consecutive BRC cells, the management layer can calculate the following:

- The difference between two TUCs (Nt)
- The difference between two TRCCs (Nr)

Information about the connection can be gained by comparing Nt and Nr:

- If Nt > Nr, the difference indicates the number of lost cells of this block test.
- If Nt < Nr, the difference indicates the number of misinserted cells of this block test.

- When Nt = Nr, no cells are lost or misinserted.

35.7 User-Defined Cells (UDC)

Typical ATM cells are 53 bytes long and consist of a 4-byte header, 1-byte HEC, and 48-byte payload. The MPC8560 also supports user-defined cells with up to 12 bytes of extra header fields for internal information for switching applications. This choice is made during initialization by writing to the FPSMR; see Section 35.13.2, “FCC Protocol-Specific Mode Register (FPSMR).” As shown in Figure 35-19, the extra header size can vary between 1 to 12 bytes (byte resolution) and the HEC octet is optional.

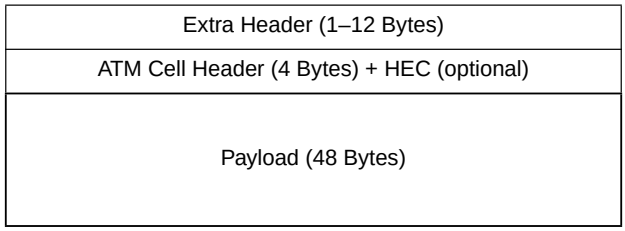


Figure 35-19. Format of User-Defined Cells

For AAL5 and AAL1 the extra header is taken from the Rx and Tx BDs. The transmitter reads the extra header from the UDC TxBD and adds it to each ATM cell associated with the current buffer. At the receive side, the extra header of the last cell in the current buffer is written to the UDC RxBD.

For AAL0 the extra header is attached to the regular ATM cell in the buffer. The transmitter reads the extra header and the ATM cell from the buffer. The receiver writes the extra header and the regular ATM cell to the buffer.

35.7.1 UDC Extended Address Mode (UEAD)

For external CAM accesses, the UDC extra header can be used to supply extra routing information; see Figure 35-20. If GMODE[UEAD] = 1, two bytes of the UDC header are used as extensions to the ATM address and the CAM match cycle performs a double-word access. UEAD_OFFSET in the parameter RAM determines the offset from the beginning of the UDC extra header to the UEAD entry. The offset should be half-word aligned (even address). See Section 35.10.1, “Parameter RAM.”

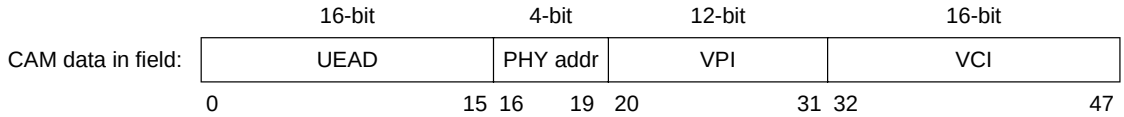


Figure 35-20. External CAM Address in UDC Extended Address Mode

35.8 ATM Layer Statistics

ATM layer statistics can be used to identify problems, such as the line-bit error rate, that affect the UNI performance. Statistics are kept in three 16-bit wrap-around counters:

- UTOPIA error dropped cells count—Counts cells discarded due to UTOPIA errors: Rx parity errors and short or long cells.
- Misinserted dropped cell count—Counts cells discarded due to address look-up failure.
- CRC10 error dropped cell count—Counts cells discarded due to CRC10 errors. (ABR only).

Counters are implemented in the dual-port RAM for each PHY device. The counters of each PHY are located in the UNI statistics table, described in Section 35.10.7, “UNI Statistics Table.”

35.9 ATM-to-TDM Interworking

The MPC8560 supports ATM and TDM interworking. The MCCs and their corresponding SIs handle the TDM data processing. (See Chapter 33, “Multi-Channel Controllers (MCCs),” and Chapter 22, “Serial Interface with Time-Slot Assigner.”) The ATM controller processes the ATM data.

Possible interworking applications include the following:

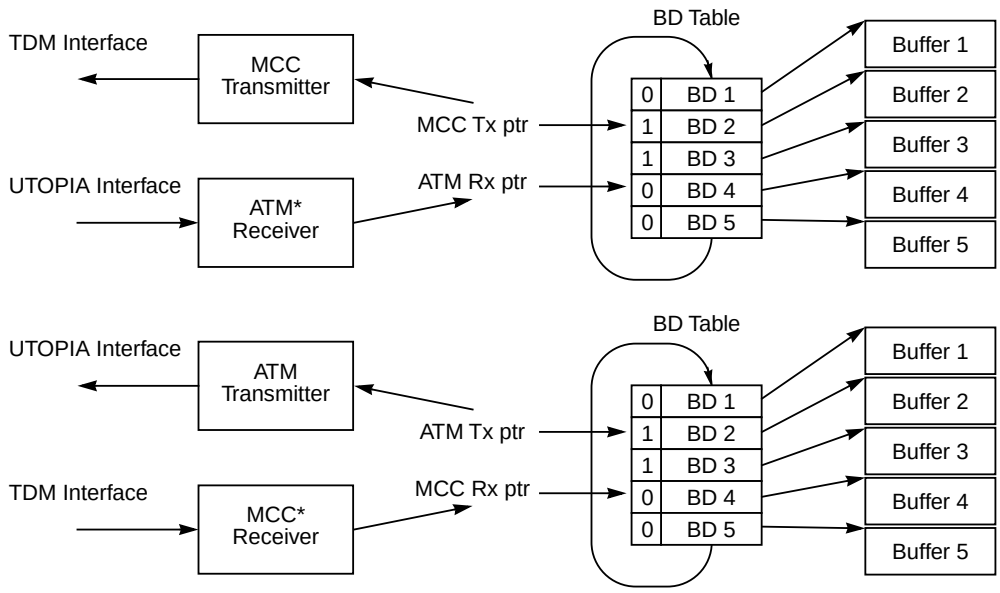
- Circuit emulation service (CES)
- Carrying voice over ATM
- Multiplexing several low speed services, such as voice and data, onto one ATM connection

Data forwarding between the ATM controller and an MCC can be done in two ways:

- Core intervention. When an MCC receive buffer is full and its RxBD is closed, the MCC interrupts the core. The core copies the MCC’s receive buffer pointer to an ATM TxBD and sets the ready bit (TxBD[R]). Similarly, when an ATM receive buffer is full and its RxBD is closed, the core services the ATM controller’s interrupt by copying the ATM receive buffer pointer to an MCC TxBD and setting TxBD[R]. This mode is useful when additional core processing is required.
- Automatic data forwarding. This mode enables automatic data forwarding between AAL1/AAL0 and transparent mode over a TDM interface.

35.9.1 Automatic Data Forwarding

The basic concept of automatic data forwarding is to program the ATM controller and the MCC to process the same BD table, as shown in Figure 35-21.



* The MCC and ATM receivers should be programmed to operate in opposite polarity E (empty) bit.

Figure 35-21. ATM-to-TDM Interworking

When going from TDM to ATM, the MCC receiver routes data from the TDM line to a specific BD table. The ATM controller transmitter is programmed to operate on the same table. When the MCC fills a receive buffer, the ATM controller sends it. The two controllers synchronize on the MCC's RxBD[E] and the ATM controller's TxBD[R].

When going from ATM to TDM, the ATM receiver reassembles data received from a particular channel to a specific BD table. The MCC transmitter is programmed to operate on the same table. When the ATM controller fills a receive buffer, the MCC controller sends it. The controllers synchronize on the ATM controller's RxBD[E] and the MCC's TxBD[R].

The MCC and ATM receivers must be programmed to operate in opposite E-bit polarity. That is, both receivers receive data into buffers whose RxBD[E] = 0 and set RxBD[E] when a buffer is full. For the ATM receiver, set RCT[INVE] of the AAL1- and AAL0-specific areas of the receive connection table; see Section 35.10.2.2, "Receive Connection Table (RCT)." For the MCC receiver, set CHAMR[EP]; see Section 33.8.1, "Channel Mode Register (CHAMR)—Transparent Mode."

35.9.2 Using Interrupts in Automatic Data Forwarding

The core can program the MCC and ATM interrupt mechanism to trigger interrupts for events such as a buffer closing or transfer errors. The interrupt mechanism can be used to synchronize the start of the automatic bridging process. For example, to start the MCC transmitter after a specific buffer reaches the ATM receiver (the buffering is required to cope with the ATM network's CDV), set ATM RxBD[I]. When the receive buffer is full, the

RxBD is closed, RxBD[E] is set (because it is operating in opposite E-bit polarity), and the core is interrupted. The core then starts the MCC transmitter.

35.9.3 Timing Issues

Use of the TDM interface assumes that all communicating entities are synchronized (that is, that they are using a synchronized serial clock). If the TDM interfaces are not synchronized, a slip can occur in the reassembly buffer. If a buffer-not-ready event occurs at the MCC transmitter, the user must restart the MCC transmit channel. If a buffer-not-ready event occurs at the ATM transmitter, the user must restart the ATM transmit channel.

35.9.4 Clock Synchronization (SRTS and Adaptive FIFOs)

Clock synchronization methods, such as using a time stamp (SRTS) or adaptive FIFOs, prevent buffer slipping during reassembly. The SRTS method may be implemented using external logic. The MPC8560 can read the SRTS from external logic and insert it into AAL1 cells, and can track the SRTS from AAL1 cells and deliver it to external logic. See Section 35.15, “SRTS Generation and Clock Recovery Using External Logic.”

Alternatively, an adaptive FIFOs method can be implemented using the core to maintain the bridging buffer at a mid-level point. The difference between the MCC and ATM data pointers is a measure of buffer synchronization. The core calculates the difference between pointers at regular intervals and adapts the TDM clock accordingly to hold the difference constant.

35.9.5 Mapping TDM Time Slots to VCs

Using the MCC and the SI, any TDM time-slot combination can be routed to a specific data buffer. (See Chapter 33, “Multi-Channel Controllers (MCCs),” and Chapter 22, “Serial Interface with Time-Slot Assigner.”) The same data buffers should be used by the ATM controller to route receive and transmit data. For information about ATM buffers see Section 35.10.5, “ATM Controller Buffer Descriptors (BDs).”

35.9.6 CAS Support

For applications requiring channel-associated signaling (CAS), circuit emulation with CAS requires additional core processing. External framers perform the CAS manipulation through a serial or parallel interface.

When the MCC receives a multi-frame block, it generates an interrupt to the core. The core reads the CAS block from the external framer and places it at the end of the ATM data buffer after the structured multi-frame block. The core then passes the buffer pointer to the ATM

controller, and the controller packs the data and CAS block into AAL1 cells. All AAL1 functions, such as generating PDU-headers and structured pointers, operate normally.

When the ATM controller receives a multi-frame block, it generates an interrupt to the core. The core reads the CAS block from the data buffer and writes it to the external framer. The core then moves the buffer pointer to the MCC. The buffer's data length should not include the CAS octets.

To optimize the process, the framer may interrupt the core only when the CAS information changes. (CAS information changes slowly.) The core can keep the CAS block in memory and connect to the framer only when the CAS changes. The core can use regular read and write cycles when connecting to the framer through a parallel interface.

The MCC and ATM controller should be synchronized with the framer's multi-frame block boundary. At the ATM side, the structured block size should equal the multi-frame block size plus the size of the CAS block so that the structured pointer, inserted by the ATM controller, points to the start of the structured data block. At the MCC side, the MCC must to be synchronized with the super frame sync signal. This synchronization can be achieved by external logic that triggers on the super frame sync signal and starts delivering the frame sync to the MCC. When loss of super frame synchronization occurs, this logic should reset and trigger again on the next super frame indication.

35.9.7 Trunk Condition

According to the Bellcore standard, the interworking function (IWF) should be able to transmit special payload on both ATM and TDM channels to signal alarm conditions (Bellcore TR-NWT-000170). The core can be used to generate the trunk condition payload in special buffers (or existing buffers) for the ATM controller or MCC.

35.9.8 ATM-to-ATM Data Forwarding

Automatic data forwarding can be used to switch ATM AAL0 cells from one ATM port to another without core intervention. The ATM receiver and transmitter should be programed to process the same BD table. When the ATM receiver fills an AAL0 buffer, the ATM transmitter sends it. The ATM receiver and transmitter are synchronized using the same mechanism as described for ATM-to-TDM automatic forwarding; see Section 35.9.1, "Automatic Data Forwarding."

35.10 ATM Memory Structure

The ATM memory structure, described in the following sections, includes the parameter RAM, the connection tables, OAM performance monitoring tables, the APC data structure, BD tables, the AAL1 sequence number protection table and the UNI statistics table.

35.10.1 Parameter RAM

When configured for ATM mode, the FCC parameter RAM is mapped as shown in Table 35-11.

Table 35-11. ATM Parameter RAM Map

Offset [1]	Name	Width	Description
0x00–0x3F	—	—	Reserved, should be cleared.
0x40	RCELL_TMP_BASE	Hword	Rx cell temporary base address. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64 byte aligned. User-defined offset from dual-port RAM base. (Recommended address space: 0x3000-0x4000 or 0xB000–0xC000)
0x42	TCELL_TMP_BASE	Hword	Tx cell temporary base address. Points to total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User-defined offset from dual-port RAM base. (Recommended address space: 0x3000–0x4000 or 0xB000–0xC000)
0x44	UDC_TMP_BASE	Hword	UDC mode only. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User-defined offset from dual-port RAM base. (Recommended address space: 0x3000–0x4000 or 0xB000–0xC000)
0x46	INT_RCT_BASE	Hword	Internal receive connection table base. User-defined offset from dual-port RAM base.
0x48	INT_TCT_BASE	Hword	Internal transmit connection table base. User-defined offset from dual-port RAM base.
0x4A	INT_TCTE_BASE	Hword	Internal transmit connection table extension base. User-defined offset from dual-port RAM base.
0x4C	—	Word	Reserved, should be cleared.
0x50	EXT_RCT_BASE	Word	External receive connection table base. User-defined.
0x54	EXT_TCT_BASE	Word	External transmit connection table base. User-defined.
0x58	EXT_TCTE_BASE	Word	External transmit connection table extension base. User-defined.
0x5C	UEAD_OFFSET	Hword	User-defined cells mode only. Offset to the user-defined extended address (UEAD) in the UDC extra header. Must be an even address. See Section 35.10.1.1, “Determining UEAD_OFFSET (UEAD Mode Only).” If RCT[BO] = 01, UEAD_OFFSET should be in little-endian format. For example, if the UEAD entry is the first half word of the extra header in external memory, UEAD_OFFSET should be programmed to 2 (second half word entry in dual-port RAM).
0x5E	—	Hword	Reserved, should be cleared.
0x60	PMT_BASE	Hword	Performance monitoring table base. User-defined offset from dual-port RAM base.
0x62	APCP_BASE	Hword	APC parameter table base address. User-defined offset from dual-port RAM base.
0x64	FBT_BASE	Hword	Free buffer pool parameter table base. User-defined offset from dual-port RAM base.

Table 35-11. ATM Parameter RAM Map (continued)

Offset [1]	Name	Width	Description
0x66	INTT_BASE	Hword	Interrupt queue parameter table base. User-defined offset from dual-port RAM base.
0x68	—	—	Reserved, should be cleared.
0x6A	UNI_STATT_BASE	Hword	UNI statistics table base. User-defined offset from dual-port RAM base. Note that this must be set up according to Section 29.10.7, “UNI Statistics Table>” It is not optional.
0x6C	BD_BASE_EXT	Word	BD table base address extension. BD_BASE_EXT[0–7] holds the 8 most-significant bits of the Rx/Tx BD table base address. BD_BASE_EXT[8–31] should be zero. User-defined.
0x70	VPT_BASE / EXT_CAM_BASE	Word	Base address of the address compression VP table/external CAM. User-defined.
0x74	VCT_BASE	Word	Base address of the address compression VC table. User-defined.
0x78	VPT1_BASE / EXT_CAM1_BASE	Word	Base address of the address compression VP1 table/EXT CAM1. User-defined.
0x7C	VCT1_BASE	Word	Base address of the address compression VC1 table. User-defined.
0x80	VP_MASK	Hword	VP mask for address compression lookup. User-defined.
0x82	VCIF	Hword	VCI filtering enable bits. When cells with VCI = 3, 4, 6, 7-15 are received and the associated VCIF bit = 1 the cell is sent to the raw cell queue. VCIF[0–2, 5] should be zero. See Section 35.10.1.2, “VCI Filtering (VCIF).”
0x84	GMODE	Hword	Global mode. User-defined. See Section 35.10.1.3, “Global Mode Entry (GMODE).”
0x86	COMM_INFO	Hword	The information field associated with the last host command. User-defined. See Section 35.14, “ATM Transmit Command.”
0x88		Hword	
0x8A		Hword	
0x8C	—	Word	Reserved, should be cleared.
0x90	CRC32_PRES	Word	Preset for CRC32. Initialize to 0xFFFF_FFFF.
0x94	CRC32_MASK	Word	Constant mask for CRC32. Initialize to 0xDEBB_20E3.
0x98	AAL1_SNPT_BASE	Hword	AAL1 SNP protection look-up table base address. (AAL1 only.) The 32-byte table resides in dual-port RAM. AAL1_SNPT_BASE must be halfword-aligned. User-defined offset from dual-port RAM base. See Section 35.10.6, “AAL1 Sequence Number (SN) Protection Table (AAL1 Only).”
0x9A	—	Hword	Reserved, should be cleared.
0x9C	SRTS_BASE	Word	External SRTS logic base address. AAL1 only. Should be 16-byte aligned. The four least-significant bits are taken from SRTS_DEV in the AAL1-specific area of the connection table entries.
0xA0	IDLE/UNASSIGN_BASE	Hword	Idle/unassign cell base address. Points to dual-port RAM area contains idle/unassign cell template (little-endian format). Should be 64-byte aligned. User-defined offset from dual-port RAM base. The ATM header should be 0x0000_0000 or 0x0100_0000 (CLP=1).

Table 35-11. ATM Parameter RAM Map (continued)

Offset [1]	Name	Width	Description
0xA2	IDLE/UNASSIGN_SIZE	Hword	Idle/unassign cell size. 52 in regular mode; 53–64 in UDC mode.
0xA4	EPAYLOAD	Word	Reserved payload. Initialize to 0x6A6A_6A6A.
0xA8	Trm	Word	(ABR only) The upper bound on the time between F-RM cells for an active source. TM 4.0 defines the Trm period as 100 msec. The Trm value is defined by the system clock and the time stamp timer prescaler; see Section 20.3.7, “RISC Time-Stamp Control Register (RTSCR).” For time stamp prescalar of 1µs, program Trm to be 100 ms/1µs = 100,000.
0xAC	Nrm	Hword	(ABR only) Controls the maximum cells the source may send for each F-RM cell. Set to 32 cells.
0xAE	Mrm	Hword	(ABR only) Controls the bandwidth between F-RM, B-RM and user data cell. Set to 2 cells.
0xB0	TCR	Hword	(ABR only) Tag cell rate. The minimum cell rate allowed for all ABR channels. An ABR channel whose ACR is less than TCR sends only out-of-rate F-RM cells at TCR. Should be set to 10 cells/sec as defined in the TM 4.0. Uses the ATMF TM 4.0 floating-point format. Note that the APC minimum cell rate (MCR) should be at least TCR.
0xB2	ABR_RX_TCTE	Hword	(ABR only) Points to total of 16 bytes reserved dual-port RAM area used by the CP. Should be 16-byte aligned. User-defined offset from dual-port RAM base.

1 Offset from FCC base: 0x8_8400 (FCC1) and 0x8_8500 (FCC2); see Section 20.5.2, “Parameter RAM.”

35.10.1.1 Determining UEAD_OFFSET (UEAD Mode Only)

The UEAD_OFFSET value is based on the position of the user-defined extended address (UEAD) in the UDC extra header. Table 35-12 shows how to determine UEAD_OFFSET: first determine the halfword-aligned location of the UEAD, and then read the corresponding UEAD_OFFSET value.

Bits/ Header Offset	0–15	16–31
0x0	UEAD_OFFSET = 0x2	UEAD_OFFSET = 0x0
0x4	UEAD_OFFSET = 0x6	UEAD_OFFSET = 0x4
0x8	UEAD_OFFSET = 0xA	UEAD_OFFSET = 0x8

Table 35-12. UEAD_OFFSETs for Extended Addresses in the UDC Extra Header

35.10.1.2 VCI Filtering (VCIF)

VCI filtering enable bits are shown in Figure 35-22.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	0	0	0	VC3	VC4	0	VC6	VC7	VC8	VC9	VC10	VC11	VC12	VC13	VC14	VC15

Figure 35-22. VCI Filtering Enable Bits

Figure 35-13 describes the operation of the VCI filtering enable bits.

Table 35-13. VCI Filtering Enable Field Descriptions

Bits	Name	Description
0–2, 5	—	Clear these bits.
3, 4, 6, 7–15	VC*x*	VCI filtering enable 0 Do not send cells with this VCI to the raw cell queue. 1 Send cells with this VCI to the raw cell queue.

35.10.1.3 Global Mode Entry (GMODE)

Figure 35-23 shows the layout of the global mode entry (GMODE).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	0	0	0	0	0	0	ALB	CTB	REM	0	0	UEAD	CUAB	EVPT	0	ALM

Figure 35-23. Global Mode Entry (GMODE)

Table 35-14 describes GMODE fields.

Table 35-14. GMODE Field Descriptions

Bits	Name	Description
0–5	—	Reserved, should be cleared.
6	ALB	Address look up bus for address compression tables. If CAMs are used and no compression tables are used, this bit should be set to 1 for improved performance. 0 Reside on the system bus. Note that it is not applicable to CAMs. 1 Reside on the local bus. For performance reasons address compression tables should reside on local bus, as the local bus has been optimized to support small transfer sizes and provide low latency access to the CPM. CAMs can only be connected to local bus.
7	CTB	External connection tables bus 0 Reside on the system bus. 1 Reside on the local bus.
8	REM	Receive emergency mode 0 Enable REM operation. When the receive FIFO is full, the ATM transmitter stops sending data cells until the receiver emergency state is cleared (FIFO not full). The transmitter pace is maintained, although a small CDV may be introduced. This mode enables the receiver to receive bursts of cells above the steady state performance. 1 Disable REM operation. Note that to check system performance the user may want to set this bit.
9–10	—	Reserved, should be cleared.
11	UEAD	User-defined cells extended address mode. See Section 35.7.1, “UDC Extended Address Mode (UEAD).” 0 Disable UEAD mode. 1 Enable UEAD mode.
12	CUAB	Check unallocated bits 0 Do not check unallocated bits during address compression. 1 Check unallocated bits during address compression.

Table 35-14. GMODE Field Descriptions (continued)

Bits	Name	Description
13	EVPT	External address compression VP table 0 VP table resides in dual-port RAM. 1 VP table reside in external memory.
14	—	Reserved, should be cleared.
15	ALM	Address look-up mechanism. See Section 35.4, “VCI/VPI Address Lookup Mechanism.” 0 External CAM lookup. 1 Address compression.

35.10.2 Connection Tables (RCT, TCT, and TCTE)

The receive and transmit connection tables, RCT and TCT, store host-initialized connection parameters after connection set-up. These include AAL type, connection traffic parameters, BD parameters and temporary parameters used during segmentation and reassembly (SAR). The transmit connection table extension (TCTE) supports special connections that use ABR, VBR or UBR+ services. Each connection table entry resides in a 32-byte space. Table 35-15 lists sizes for RCT, TCT, and TCTE.

Table 35-15. Receive and Transmit Connection Table Sizes

ATM Service Class	RCT	TCT	TCTE
CBR, UBR service	32 bytes	32 bytes	—
ABR, VBR, UBR+ service	32 bytes	32 bytes	32 bytes

Note that an ATM channel is considered internal if its tables are in an internal dual-port RAM; it is considered external if its tables are in external memory.

NOTE

To improve performance, store parameters for fast channels in internal dual-port RAM and parameters for slower channels in external memory. Connection tables for external channels are read and written from external memory each time the CP processes a cell. As the local bus has been optimized to provide low latency access to the CPM, connection tables for external connections should reside on the local bus to improve performance. The CP does, however, minimize memory access time by burst fetching the 32-byte entry and writing back only the first 24 bytes.

In all connection tables, fields which are not used must be cleared.

35.10.2.1 ATM Channel Code

Each ATM channel has a channel code used as an index to the channel's connection table entry. The first channel in the table has channel code one, the second has channel code two, and so on. Codes of 255 or less indicate internal channels; codes greater than 255 indicate external channels. Channel code one is reserved as the raw cell queue and cannot be used for another purpose. The channel code is used to specify a VC when sending a ATM TRANSMIT command, initiating the external CAM or address compression tables, and when the CP sends an interrupt to an interrupt queue.

Example:

Suppose a configuration supports 1,024 regular ATM channels. To allocate 4 Kbytes of dual-port RAM space to the internal connection table, determine that channel codes 0–63 are internal (64 VCs × 64 bytes (RCT and TCT) = 4 K). Channels 0–1 are reserved. The remaining 962 (1024 - 62) external channels are assigned channel codes 256–1217. See Figure 35-24.

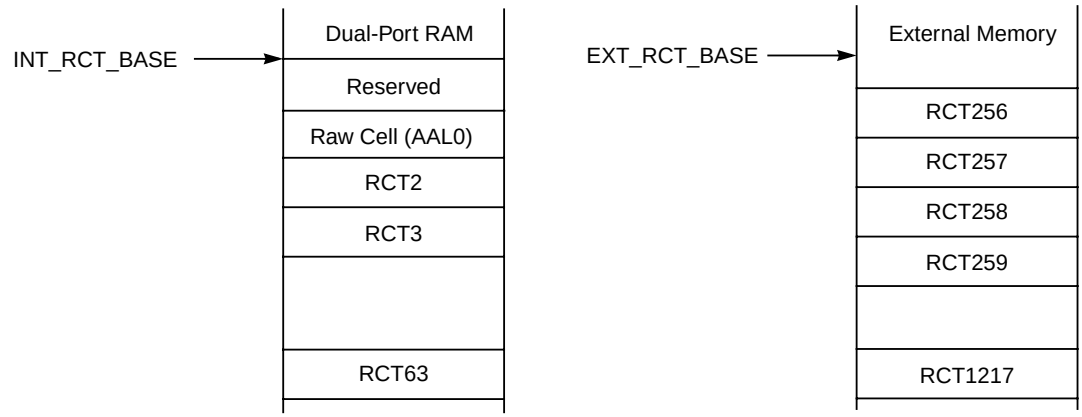


Figure 35-24. Example of a 1024-Entry Receive Connection Table

The general formula for determining the real starting address for all internal and external connection table entries is as follows:

connection table base address + (channel code × 32)

Thus, the real starting address of the RCT entry associated with channel code 3 is as follows:

INT_RCT_BASE+ (3 × 32) = INT_RCT_BASE + 96

Even though it produces a gap in the connection table, the first external channel's real starting address of the RCT entry (channel code 256) is as follows:

EXT_RCT_BASE+ (256 × 32) = EXT_RCT_BASE + 8192

See Section 35.10.1, “Parameter RAM,” to find all the connection table base address parameters. (The transmit connection table base address parameters are INT_TCT_BASE, EXT_TCT_BASE, INT_TCTE_BASE, and EXT_TCTE_BASE.)

35.10.2.2 Receive Connection Table (RCT)

Figure 35-25 shows the format of an RCT entry.

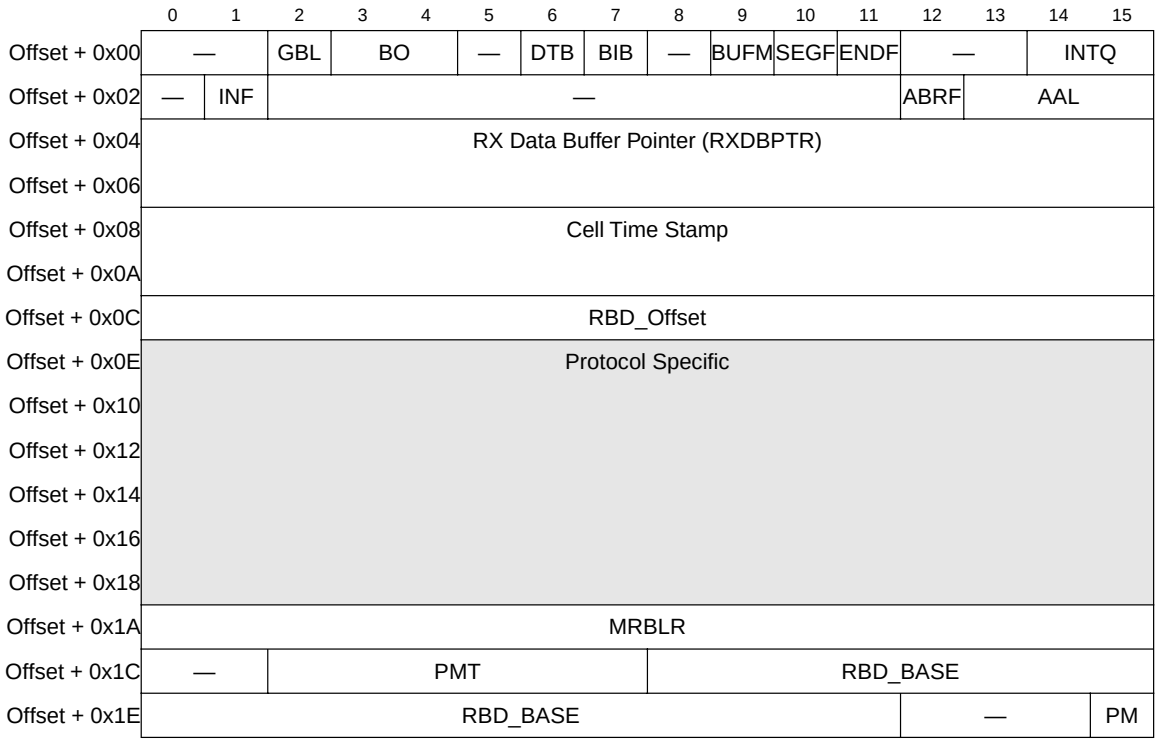


Figure 35-25. Receive Connection Table (RCT) Entry

Table 35-16 describes RCT fields.

Table 35-16. RCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared.
	2	GBL	Global. Asserting GBL enables snooping of data buffers, BD, interrupt queues and free buffer pool.
	3–4	BO	Byte ordering—used for data buffers. 0x Reserved 1x Big endian
	5	—	Reserved, should be cleared.
	6	DTB	Data buffers bus 0 Data buffers reside on the system bus. 1 Data buffers reside on the local bus.
	7	BIB	BD, interrupt queues, and free buffer pool 0 Reside on the system bus. 1 Reside on the local bus. Note that when using AAL5 or AAL1 in UDC mode, BDs must be placed on the same bus (RCT[DTB] = RCT[BIB]). This is necessary because in UDC mode the user-defined header, which is part of the cell data, is read using the same bus configuration (byte ordering and bus type) as the payload. Therefore, if data is placed on the 60x bus and the BD on the local bus, the SDMA accesses the UDC header on the 60x bus with the address of the local bus.
	8	—	Reserved, should be cleared.
	9	BUFM	Buffer mode. (AAL5 only) See Section 35.10.5.3, “ATM Controller Buffers.” 0 Static buffer allocation mode. Each BD is associated with a dedicated buffer. 1 Global buffer allocation mode. Free buffers are fetched from global free buffer pools.
	10	SEGF	OAM F5 segment filtering 0 Do not send cells with PTI = 100 to the raw cell queue. 1 Send cells with PTI = 100 to the raw cell queue.
	11	ENDF	OAM F5 end-to-end filtering 0 Do not send cells with PTI=101 to the raw cell queue. 1 Send cells with PTI=101 to the raw cell queue.
	12–13	—	Reserved, should be cleared.
	14–15	INTQ	Points to one of four interrupt queues available.
0x02	0	—	Internal use only. Should be cleared.
	1	INF	(AAL5 only) Indicates the receiver state. Initialize to 0 0 In idle state. 1 In AAL5 frame reception state.
	2–11	—	Internal use only. Should be cleared.
	12	ABRF	(AAL5 only). Controls ABR flow. 0 ABR flow control is disabled. 1 ABR flow control is enabled.
	13–15	AAL	AAL type 000 AAL0—Reassembly with no adaptation layer 001 AAL1—ATM adaptation layer 1 protocol 010 AAL5—ATM adaptation layer 5 protocol All others reserved.

Table 35-16. RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x04	—	RxDBPTR	Receive data buffer pointer. Holds real address of current position in the Rx buffer.
0x08	—	Cell Time Stamp	Used for reassembly time-out. Whenever a cell is received, the MPC8560 time stamp timer is sampled and written to this field. See Section 20.3.7, “RISC Time-Stamp Control Register (RTSCR).”
0x0C	—	RBD_Offset	RxBD offset from RBD_BASE. Points to the channel’s current BD. User-initialized to 0; updated by the CP.
0x0E-0x18		—	Protocol-specific area.
0x1A	—	MRBLR	Maximum receive buffer length. Used in both static and dynamic buffer allocation.
0x1C	0–1	—	Reserved, should be cleared.
	2–7	PMT	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE+PMT × 32. Can be changed on-the-fly.
	8–15	RBD_BASE	RxBD base. Points to the first BD in the channel’s RxBD table. The 8 most-significant bits of the address are taken from BD_BASE_EXT in the parameter RAM. The four least-significant bits of the address are taken as zeros.
0x1E	0–11		
	12–14	—	Reserved, should be cleared.
	15	PM	Performance monitoring. Can be changed on-the-fly. 0 No performance monitoring for this VC. 1 Perform performance monitoring for this VC. Whenever a cell is received for this VC the performance monitoring table that its code is written in the PMT field is updated.

35.10.2.2.1 AAL5 Protocol-Specific RCT

Figure 35-26 shows the AAL5 protocol-specific area of an RCT entry.

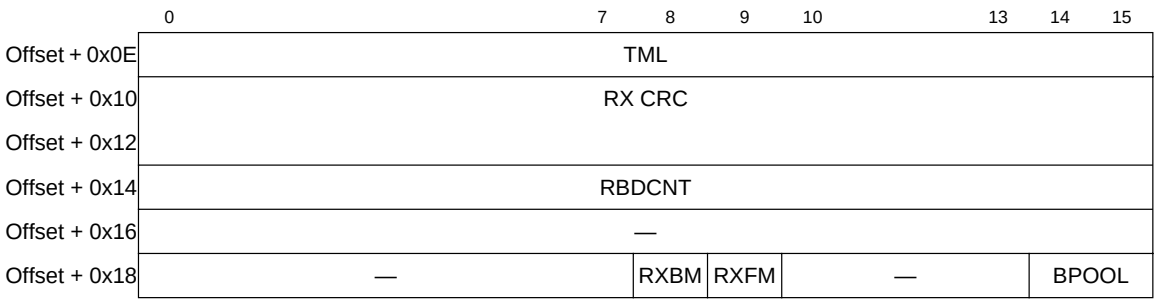


Figure 35-26. AAL5 Protocol-Specific RCT

Table 35-17 describes AAL5 protocol specific RCT fields.

Table 35-17. RCT Settings (AAL5 Protocol-Specific)

Offset	Bits	Name	Description
0x0E	—	TML	Total message length. This field is used by the CP.
0x10	—	RxCRC	CRC32 temporary result.
0x14	—	RBDCNT	RxBD count. Indicates how may bytes remain in the current Rx buffer. RBDCNT is initialized with MRBLR whenever the CP opens a new buffer.
0x16	—	—	Reserved, should be cleared.
0x18	0–7	—	Reserved, should be cleared.
	8	RXBM	Receive buffer interrupt mask. Determines whether the receive buffer event is disabled. Can be changed on-the-fly. 0 The event is disabled for this channel. (The RXB event is not sent to the interrupt queue when receive buffers are closed.) 1 The event is enabled for this channel.
	9	RXFM	Receive frame interrupt mask. Determines whether the receive frame event is disabled. Can be changed on-the-fly. 0 The event is disabled for this channel. (RXF event is not sent to the interrupt queue.) 1 The event is enabled for this channel.
	10–13	—	Reserved, should be cleared.
	14–15	BPOOL	Buffer pool. Global buffer allocation mode only. Points to one of four free buffer pools. See Section 35.10.5.2.4, “Free Buffer Pool Parameter Tables.”

35.10.2.2.2 AAL5-ABR Protocol-Specific RCT

Figure 35-27 shows the AAL5-ABR protocol-specific area of an RCT entry.

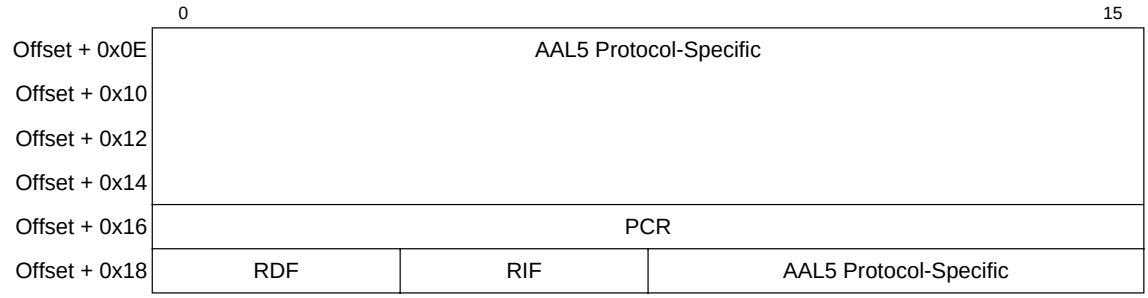


Figure 35-27. AAL5-ABR Protocol-Specific RCT

Table 35-18 describes AAL5-ABR protocol-specific RCT fields.

Table 35-18. ABR Protocol-Specific RCT Field Descriptions

Offset	Bits	Name	Description
0x0E	—	—	AAL5 protocol-specific
0x16	—	PCR	Peak cell rate. The peak number of cells per second of the current ABR channel. The ACR (allowed cell rate) never exceeds this value. PCR uses the ATMF TM 4.0 floating-point format.

Table 35-18. ABR Protocol-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x18	0–3	RDF	Rate decrease factor for the current ABR channel. Controls the decrease in cell transmission rate upon receipt of a backward RM cell. RDF represents a negative exponent of two, that is, the decrease factor = 2^{-RDF}. The decrease factor ranges from 1/32768 (RDF=0xF) to 1 (RDF=0).
	4–7	RIF	Rate increase factor of the current ABR channel. Controls the increase in the cell transmission rate upon receipt of a backward RM cell. RIF represents a negative exponent of two, that is, the increase factor = 2^{-RIF}. The increase factor ranges from 1/32768 (RIF=0xF) to 1 (RIF=0).
	8–15	—	AAL5 protocol-specific

35.10.2.2.3 AAL1 Protocol-Specific RCT

Figure 35-28 shows the AAL1 protocol-specific area of an RCT entry.

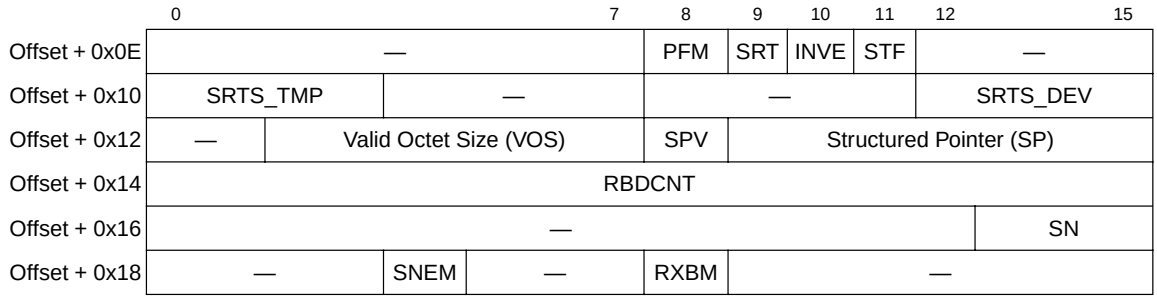


Figure 35-28. AAL1 Protocol-Specific RCT

Table 35-19 describes AAL1 protocol-specific RCT fields.

Table 35-19. AAL1 Protocol-Specific RCT Field Descriptions

Offset	Bits	Name	Description
0x0E	0–7	—	Reserved, should be cleared.
	8	PFM	Partially filled mode. 0 Partially filled cells mode is not used. 1 Partially filled cells mode is used. The receiver copies only valid octets from the AAL1 cell to the Rx buffer. The number of the valid octets from the beginning of the AAL1 user data field is specified in the VOS (valid octet size) field.
	9	SRT	Synchronous residual time stamp. Unstructured format only. The MPC8560 supports clock recovery using an external SRTS PLL. The MPC8560 tracks the SRTS from the incoming four cells with SN = 1, 3, 5, and 7 and writes it to the external SRTS device. Every eight cells the CP writes a valid SRTS to external logic. (See Section 35.15, “SRTS Generation and Clock Recovery Using External Logic.”) 0 SRTS mode is not used. 1 SRTS mode is used.
	10	INVE	Inverted empty. 0 RxBD[E] is interpreted normally (1 = empty, 0 = not empty). 1 RxBD[E] is handled in negative logic (0 = empty, 1 = not empty).
	11	STF	Structured format 0 Unstructured format is used. 1 Structured format is used.
	12–15	—	Reserved, should be cleared.
0x10	0–3	SRTS_TMP	Used by the CP to store the received SRTS code. After a cell with SN = 7 is received, the CP writes the SRTS code to the external SRTS device.
	4–11	—	Reserved, should be cleared.
	12–15	SRTS_DEV	Selects an SRTS device, whose address is SRTS_BASE[0–27] + SRTS_DEV[28–31]. The 16 byte-aligned SRTS_BASE is taken from the parameter RAM.
0x12	0–1	—	Reserved, should be cleared.
	2–7	VOS	Valid octet size. Specifies the number of valid octets from the beginning of the AAL1 user data field. For unstructured, service values 1–47 are valid; for structured service, values 1-46 are valid. Partially filled cell mode only.
	8	SPV	Structured pointer valid. Should be user-initialized user to zero. Structured format only.
	9–15	SP	Structured pointer. Used by the CP to calculate the structured pointer. This field should be initialized by the user to zero. Used in structured format only.
0x14	—	RBDCNT	RxBD count. Indicates how may bytes remain in the current Rx buffer. Initialized with MRBLR whenever the CP opens a new buffer.
0x16	0–12	—	Reserved, should be cleared.
	13–15	SN	Sequence number. Used by the CP to check incoming cell’s sequence number.

Table 35-19. AAL1 Protocol-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x18	0–3	—	Reserved, should be cleared.
	4	SNEM	Sequence number error flag interrupt mask 0 This mode is disabled. 1 When an out-of-sequence error occurs, an RXB interrupt is sent to the interrupt queue even if RCT[RXBM] is cleared. Note that this mode is the buffer error reporting mechanism during automatic data forwarding (ATM-to-TDM bridging) when no buffer processing is required (RCT[RXBM]=0).
	5–7	—	Reserved, should be cleared.
	8	RXBM	Receive buffer interrupt mask 0 The receive buffer event of this channel is disabled. (The event is not sent to the interrupt queue.) 1 The receive buffer event of this channel is enabled.
	9–15	—	Reserved, should be cleared.

35.10.2.2.4 AAL0 Protocol-Specific RCT

Figure 35-29 shows the layout for the AAL0 protocol-specific RCT.

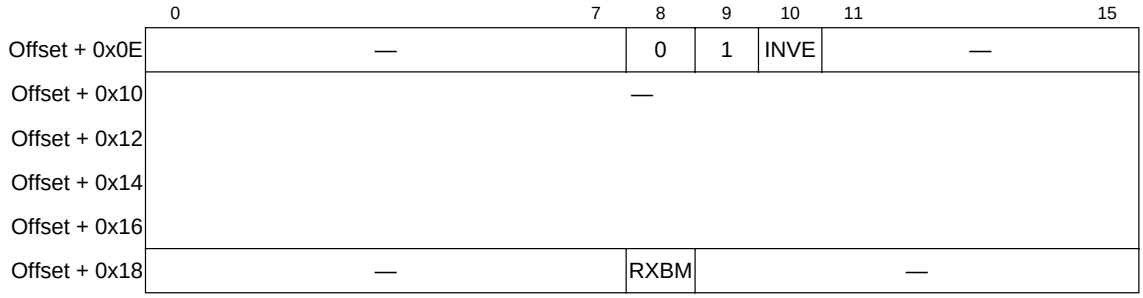


Figure 35-29. AAL0 Protocol-Specific RCT

Table 35-20 describes AAL0 protocol specific RCT fields.

Table 35-20. AAL0-Specific RCT Field Descriptions

Offset	Bits	Name	Description
0x0E	0-7	—	Reserved, should be cleared.
	8-9	0b01	Must be programmed to 0b01 for AAL0.
	10	INVE	Inverted empty. 0 RxBD[E] is interpreted normally (1 = empty, 0 = not empty). 1 RxBD[E] is handled in negative logic (0 = empty, 1 = not empty).
	11-15	—	Reserved, should be cleared.
0x10	—	—	Reserved, should be cleared.

Table 35-20. AAL0-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x18	0–7	—	Reserved, should be cleared.
	8	RXBM	Receive buffer interrupt mask 0 The receive buffer event of this channel is masked. (The RXB event is not sent to the interrupt queue when receive buffers are closed.) 1 The receive buffer event of this channel is enabled.
	9–15	—	Reserved, should be cleared.

35.10.2.3 Transmit Connection Table (TCT)

Figure 35-30 shows the format of an TCT entry.

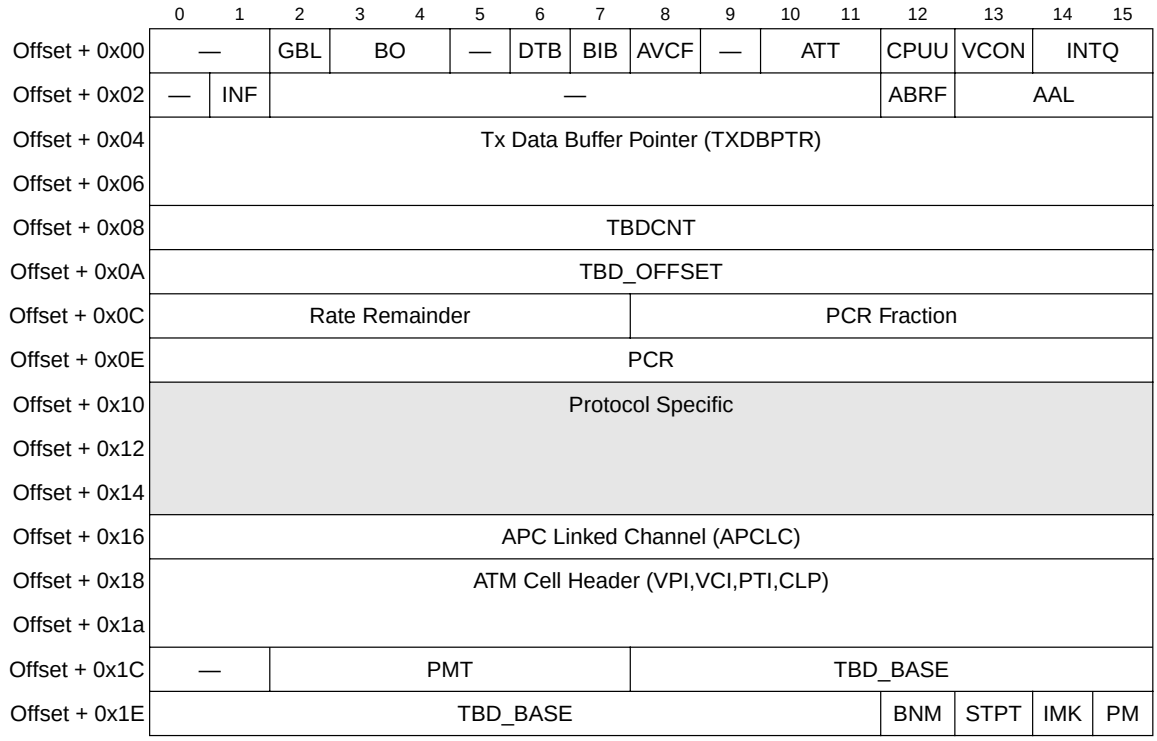


Figure 35-30. Transmit Connection Table (TCT) Entry

Table 35-21 describes general TCT fields.

Table 35-21. TCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared.
	2	GBL	Global. Asserting GBL enables snooping of data buffers, BDs, interrupt queues and free buffer pool.
	3–4	BO	Byte ordering. This field is used for data buffers. 0x Reserved 1x Big endian
	5	—	Reserved, should be cleared.
	6	DTB	Data buffer bus 0 Reside on the system bus. 1 Reside on the local bus.
	7	BIB	BD and interrupt queue 0 Reside on the system bus. 1 Reside on the local bus. Note: When using AAL5, AAL1 in UDC mode, BDs and data should be placed on the same bus (TCT[DTB]=TCT[BIB]).
	8	AVCF	Auto VC off. Determines the behavior of the APC when the last buffer associated with this VC has been sent and no more ready buffers are in the VC's TxBD table. 0 The APC does not remove this VC from the schedule table and continues to schedule it to transmit. 1 The APC removes this VC from the schedule table. To continue transmission after the host adds buffers for transmission, a new ATM TRANSMIT command is needed, which can be issued only after the CP clears TCT[VCON]. Note that when over-subscribing UBR or UBR+ channels, set AVCF so that the CPM does not become overloaded polling non-active VCs.
	9	—	Reserved, should be cleared.
	10–11	ATT	ATM traffic type 00 Peak cell-rate pacing. The host must initialize PCR and the PCR fraction. Other traffic parameters are not used. 01 Peak and sustain cell rate pacing (VBR traffic). The APC performs a continuous-state leaky bucket algorithm (GCRA) to pace the channel-sustain cell rate. The host must initialize PCR, PCR fraction, SCR, SCR fraction, and BT (burst tolerance). 10 Peak and minimum cell rate pacing (UBR+ traffic). The host must initialize PCR, PCR fraction, MCR, MCR fraction, and MDA. 11 Reserved
	12	CPUU	CPCS-UU+CPI insertion (used for AAL5 only). 0 CPCS-UU+CPI insertion disabled. The transmitter clears the CPCS-UU+CPI fields. 1 CPCS-UU+CPI insertion enabled. The transmitter reads the CPCS-UU+CPI (16-bit entry) from external memory. It should be placed after the end of the last buffer (it should not be included in the buffer length).
	13	VCON	Virtual channel is on. Should be set by the host before it issues an ATM TRANSMIT command. When the host sets TCT[STPT] (stop transmit), the CP deactivates this channel and clears VCON when the channel is next encountered in the APC scheduling table. The host can issue another ATM TRANSMIT command only after the CP clears VCON.
	14–15	INTQ	Points to one of four interrupt queues available.

Table 35-21. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	0	—	Internal use only. Should be cleared.
	1	INF	Used for AAL5 Only. Indicates the transmitter state. Initialize to 0 0 In idle state. 1 In AAL5 frame transmission state.
	2–11	—	Internal use only. Should be cleared.
	12	ABRF	Used for AAL5 Only. 0 ABR Flow control is disabled. 1 ABR Flow control is enabled.
	13–15	AAL	AAL type 000 AAL0—Segmentation without any adaptation layer. 001 AAL1—ATM adaptation layer 1 protocol. 010 AAL5—ATM adaptation layer 5 protocol.
0x04	—	TxDBPTR	Tx data buffer pointer. Holds the real address of the current position in the Tx buffer.
0x08	—	TBDCNT	Transmit BD count. Counts the remaining data to transmit in the current transmit buffer. Its initial value is loaded from the data length field of the TxBD when a new buffer is open; its value is subtracted for any transmitted cell associated with this channel.
0x0A	—	TBD_OffSet	Transmit BD offset. Holds offset from TBD_BASE of the current BD. Should be cleared initially.
0x0C	0–7	Rate Reminder	Rate remainder. Used by the APC to hold the rate remainder after adding the pace fraction to the additive channel rate. Should be cleared initially.
	8–15	PCR Fraction	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot. If this is an ABR channel, this field is automatically updated by the CP.
0x0E	—	PCR	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract. Note that for an ABR channel, the CP automatically updates PCR to the ACR value.
0x10	—	—	Protocol-specific
0x16	—	APCLC	APC linked channel. Used by the CP. Initialize to 0 (null pointer).
0x18	—	ATMCH	ATM cell header. Holds the full (4-byte) ATM cell header of the current channel. The transmitter appends ATMCH to the cell payload during transmission.

Table 35-21. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1C	0–1	—	Reserved, should be cleared.
	2–7	PMT	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE+PMT × 32. Can be changed on-the-fly.
	8–15	TBD_BASE	TxBD base. Points to the first BD in the channel's TxBD table. The 8 most-significant bits of the address are taken from BD_BASE_EXT in the parameter RAM. The four least-significant bits of the address are taken as zero.
0x1E	0–11		
	12	BNM	Buffer-not-ready interrupt mask. Can be changed on-the-fly. 0 The transmit buffer-not-ready event of this channel is masked. (TBNR event is not sent to the interrupt queue.) 1 The buffer-not-ready event of this channel is enabled.
	13	STPT	Stop transmit. Should be cleared initially. When the host sets this bit, the CP deactivates this channel and clears TCT[VCON] when the channel is next encountered in the APC scheduling table. Note that for AAL5 if STPT is set and frame transmission is already started (TCT[INF]=1), an abort indication will be sent (last cell with zero length field).
0x1E	14	IMK	Interrupt mask. Can be changed on-the-fly. 0 The transmit buffer event of this channel is masked. (TXB event is not sent to the interrupt queue.) 1 The transmit buffer event of this channel is enabled.
	15	PM	Performance monitoring. Can be changed on-the-fly. 0 No performance monitoring for this VC. 1 Performance is monitored for this VC. When a cell is sent for this VC, the performance monitoring table indicated in PMT field is updated.

35.10.2.3.1 AAL5 Protocol-Specific TCT

Figure 35-31 shows the AAL5 protocol-specific TCT.

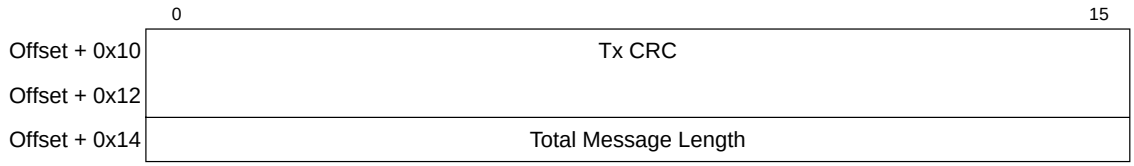


Figure 35-31. AAL5 Protocol-Specific TCT

Table 35-22 describes AAL5 protocol-specific TCT fields.

Table 35-22. AAL5-Specific TCT Field Descriptions

Offset	Name	Description
0x10	Tx CRC	CRC32 temporary result.
0x14	Total Message Length	This field is used by the CP.

35.10.2.3.2 AAL1 Protocol-Specific TCT

Figure 35-32 shows the AAL1 protocol-specific TCT.

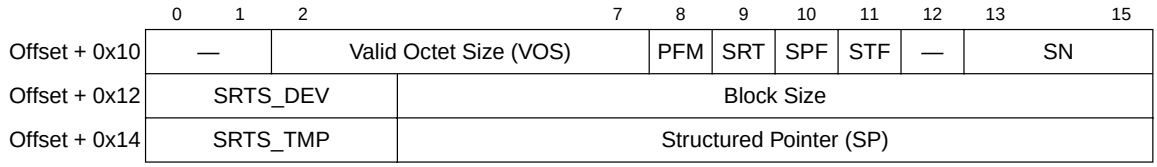


Figure 35-32. AAL1 Protocol-Specific TCT

Table 35-23 describes AAL1 protocol-specific TCT fields.

Table 35-23. AAL1-Specific TCT Field Descriptions

Offset	Bits	Name	Description
0x10	0-1	—	Reserved, should be cleared.
	2–7	VOS	Valid octet size. Partially filled cell mode only. Specifies the number of valid octets from the beginning of the AAL1 user data field. For unstructured service, values 1-47 are valid; for structured service, values 1-46 are valid.
	8	PFM	Partially filled mode. 0 Partially filled cells mode is not used. 1 Partially filled cells mode is used. The transmitter copies only valid octets from the buffer to the AAL1 cell. The size of the valid octets from the beginning of the AAL1 user data field is specified in the VOS (valid octet size) field.
	9	SRT	Synchronous residual time stamp. Unstructured format only. The MPC8560 supports SRTS generation using external logic. If this mode is enabled, the MPC8560 reads the SRTS from external logic and inserts it into four cells for which SN = 1, 3, 5, or 7. The MPC8560 reads the new SRTS from external logic every eight cells. (See Section 35.15, “SRTS Generation and Clock Recovery Using External Logic.”) 0 SRTS mode is not used. 1 SRTS mode is used.
	10	SPF	Structured pointer flag. Indicates that a structured pointer has been inserted in the current block. The user should initialize this field to zero. Used by the CP only.
	11	STF	Structured format 0 Unstructured format is used. 1 Structured format is used.
	12	—	Reserved, should be cleared.
	13–15	SN	Sequence number field. Used by the CP to check the incoming cells SN. Should be cleared initially.
0x12	0–3	SRTS_DEV	Used to select a SRTS device. The SRTS device address is SRTS_BASE[0–27]+SRTS_DEV[28:31]. SRTS_BASE is taken from the parameter RAM and is 16-byte aligned.
	4–15	Block Size	Used only in structured format. Specifies the structured block size (Block Size = 0xFFF = 4 Kbytes maximum).

Table 35-23. AAL1-Specific TCT Field Descriptions

Offset	Bits	Name	Description
0x14	0–3	SRTS_TMP	Before a cell with SN = 1 is sent, the CP reads the SRTS code from external SRTS logic, writes it to SRTS_TMP, and then inserts SRTS_TMP into the next four cells with an odd SN.
	4–15	SP	Structured pointer. Used by the CP to calculate the structured pointer. Should be cleared initially. Structured format only.

35.10.2.3.3 AAL0 Protocol-Specific TCT

Figure 35-33 shows the AAL0 protocol-specific TCT.

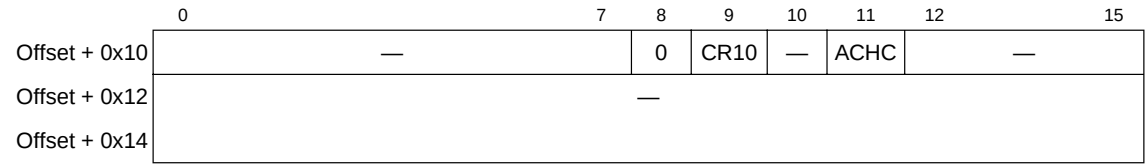


Figure 35-33. AAL0 Protocol-Specific TCT

Table 35-24 describes AAL0 protocol-specific TCT fields.

Table 35-24. AAL0-Specific TCT Field Descriptions

Offset	Bits	Name	Description
0x10	0–7	—	Reserved, should be cleared.
	8	0	Must be 0.
	9	CR10	CRC-10 0 CRC10 insertion is disabled. 1 CRC10 insertion is enabled.
	10	—	Reserved, should be cleared.
	11	ACHC	ATM cell header change 0 52 bytes are taken from the transmit buffer (all except the HEC byte). 1 28 bits are taken from TCT[ATMCH]. These 28 bits include the following fields: GFC (4 bits), VPI (8 bits), and VCI (16 bits). The remainder of the cell, except for the HEC byte, is taken from the transmit buffer. This includes the PTI/CLP (4 bits) and the 48-byte payload. Note that in the receive direction the 52 bytes are put in the receive buffer (all except the HEC byte).
	12–15	—	Reserved, should be cleared.
0x12–0x14	—	—	Reserved, should be cleared.

35.10.2.3.4 VBR Protocol-Specific TCTE

Figure 35-34 shows the VBR protocol-specific TCTE.

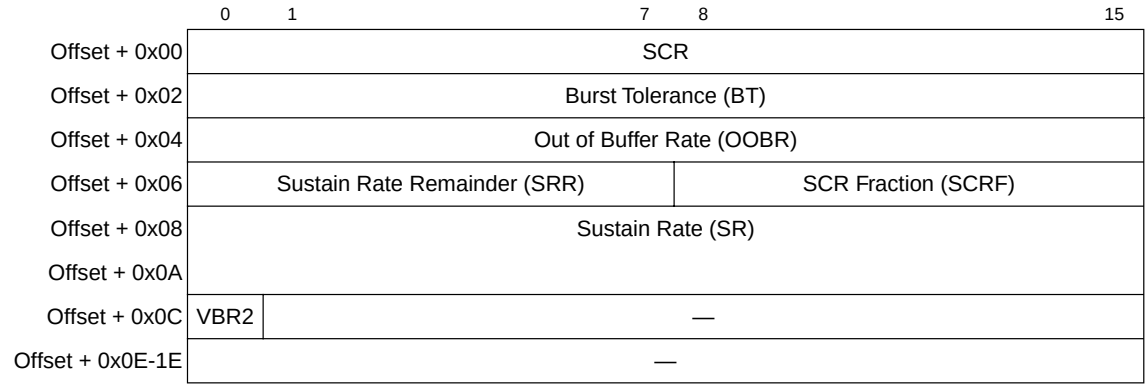


Figure 35-34. Transmit Connection Table Extension (TCTE)—VBR Protocol-Specific

Table 35-25 describes VBR protocol-specific TCTE fields.

Table 35-25. VBR-Specific TCTE Field Descriptions

Offset	Bits	Name	Description
0x00	—	SCR	Sustain cell rate. Holds the sustain cell rate (in slots) permitted for this channel according to the traffic contract. To pace the channel's sustain cell rate, the APC performs a continuous-state leaky bucket algorithm (GCRA).
0x02	—	BT	Burst tolerance. Holds the burst tolerance permitted for this channel according to the traffic contract. The relationship between the BT and the maximum burst size (MBS) is BT=(MBS-2) × (SCR-PCR) + SCR.
0x04	—	OOBR	Out-of-buffer rate. In out of buffer state (when the transmitter tries to open TxBD whose R bit is not set) the APC reschedules the current channel according to OOBR rate.
0x06	0–7	SRR	Sustain rate remainder. Holds the sustain rate remainder after adding the pace fraction field to the additive channel sustain rate. Used by the APC to calculate the channel GCRA (leaky bucket) state. Initialized to 0.
	8–15	SCRF	Holds the sustain cell rate fraction of this channel in units of 1/256 slot.
0x08	—	SR	Sustain rate. Used by the APC to hold the sustain rate after adding the pace field to the additive channel sustain rate. Used by the APC to calculate the channel GCRA (leaky bucket) state.
0x0C	0	VBR2	VBR type 0 Regular VBR. CLP=0+1 cells are rescheduled by PCR or SCR according to the GCRA state. 1 VBR Type 2. CLP=0 cells are rescheduled by PCR or SCR according to the GCRA state. CLP=1 cells are rescheduled by PCR.
	1–15	—	Reserved, should be cleared.
0x0E–0x1E	—	—	Reserved, should be cleared.

35.10.2.3.5 UBR+ Protocol-Specific TCTE

Figure 35-35 shows the UBR+ protocol-specific TCTE.

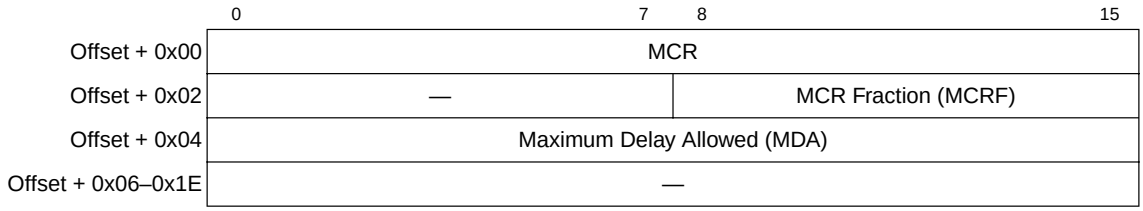


Figure 35-35. UBR+ Protocol-Specific TCTE

Table 35-26 describes UBR+ protocol-specific TCTE fields.

Table 35-26. UBR+ Protocol-Specific TCTE Field Descriptions

Offset	Bits	Name	Description
0x00	—	MCR	Minimum cell rate for this channel. MCR is in units of APC time slots.
0x02	0–7	—	Reserved, should be cleared.
	8–15	MCRF	Minimum cell rate fraction. Holds the minimum cell rate fraction of this channel in units of 1/256 slot.
0x04	—	MDA	Maximum delay allowed. The maximum time-slot service delay allowed for this priority level before the APC reduces the scheduling rate from PCR to MCR.
0x06–0x1E	—	—	Reserved, should be cleared.

35.10.2.3.6 ABR Protocol-Specific TCTE

Figure 35-36 shows the ABR protocol-specific TCTE.

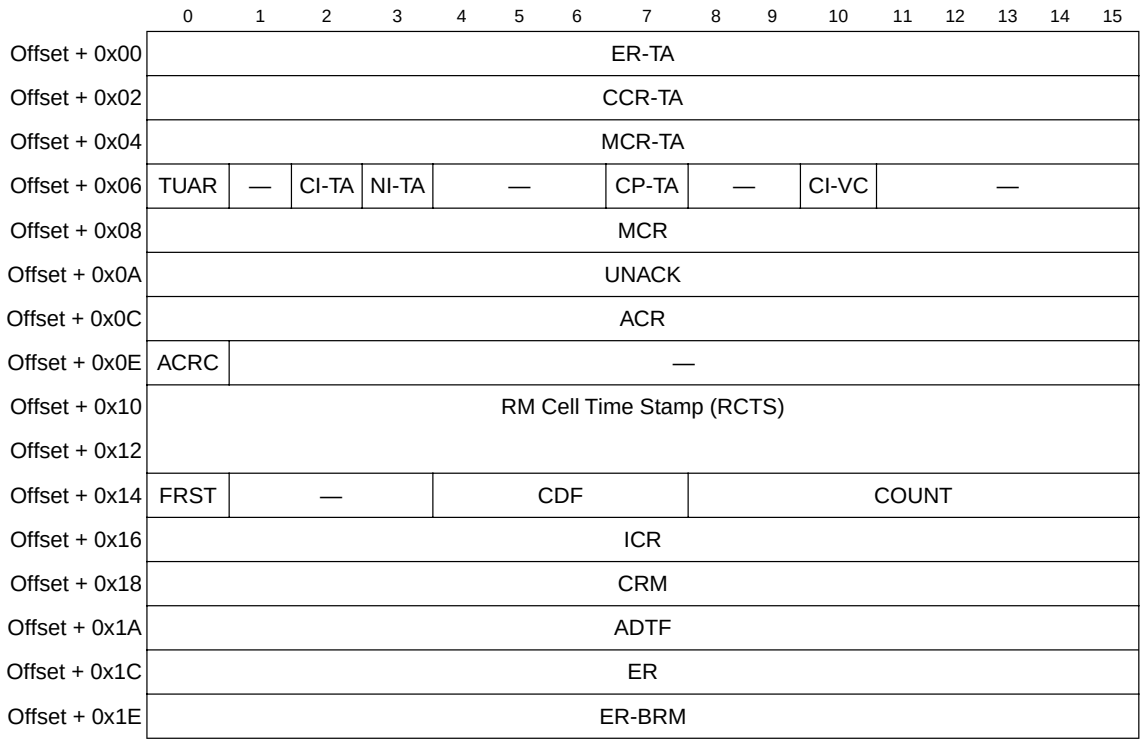


Figure 35-36. ABR Protocol-Specific TCTE

Table 35-27 describes ABR-specific TCTE fields.

Table 35-27. ABR-Specific TCTE Field Descriptions

Offset	Bits	Name	Description
0x00	—	ER-TA	Explicit rate–turn-around cell. Holds the ER of the last received F-RM cell. If another F-RM cell arrives before the previous F-RM cell was turned around, this field is overwritten by the new RM cell's ER.
0x02	—	CCR-TA	Current cell rate–turn-around cell. Holds the CCR of the last received F-RM cell. If another F-RM cell arrives before the previous F-RM cell was turned around, this field is overwritten by the new RM cell's CCR.
0x04	—	MCR-TA	Minimum cell rate–turn-around cell. Holds the MCR of the last received F-RM cell. If another F-RM cell arrives before the previous F-RM cell is turned around, this field is overwritten by the new RM cell's MCR.
0x06	0	TUAR	Turn-around flag. The CP sets TUAR to indicate that a new F-RM cell was received, which causes the transmitter to send a B-RM cell whenever the ABR flow control permits. Should be cleared initially.
	1	—	Reserved, should be cleared.
	2	CI-TA	Congestion indication–turn-around cell. Holds the CI of the last received F-RM cell. If another F-RM cell arrives before the previous F-RM cell was turned around, CI-TA is overwritten by the new RM cell's CI.

Table 35-27. ABR-Specific TCTE Field Descriptions (continued)

Offset	Bits	Name	Description
	3	NI-TA	No increase–turn-around cell. Holds the NI of the last received F-RM cell. If another F-RM cell arrives before the previous one was turned around, NI-TA is overwritten by the new RM cell's NI.
	4–6	—	Reserved, should be cleared.
	7	CP-TA	Cell loss priority–turn-around cell. Holds the CLP of the last received F-RM cell. If another F-RM cell arrives before the previous one was turned around, CP-TA is overwritten by the new RM cell's CLP.
	8–9	—	Reserved, should be cleared.
	10	CI-VC	Congestion indication -VC. Holds the EFCI (explicit forward congestion indication) of the last user data cell. The CI bit of the turned around RM cell is ORed with the CI-VC. Should be cleared initially.
	11–15	—	Reserved, should be cleared.
0x08	—	MCR	Minimum cell rate Holds the minimum number of cells/sec of the current ABR channel. Uses the ATMF TM 4.0 floating-point format.
0x0A	—	UNACK	Used by the CP to count F-RM cells sent in an absence of received B-RM cells. Should be cleared initially.
0x0C	—	ACR	Allowed cell rate The cells per second allowed for the current ABR channel. Uses the ATMF TM 4.0 floating-point format. Initialize with ICR.
0x0E	0	ACRC	ACR change. Indicates a change in ACR. Initialize to one.
	1–15	—	Reserved, should be cleared.
0x10	—	RCTS	RM cell time stamp. Used exclusively by the CP. Initialize to zero.
0x14	0	FRST	First turn. Used exclusively by the CP. Indicates the first turn of a backward RM cell, which has priority over a data cell. Initialized to 0.
	1–3	—	Reserved, should be cleared.
	4–7	CDF	Cutoff decrease factor. Controls the decrease in the ACR associated with missing B-RM cells feedback. CDF represents a negative exponent of two, that is, the cutoff decrease factor = 2^{-CDF}. The cutoff decrease factor ranges from 1/64 (CDF=0b0110) to 1 (CDF=0b0000). All other CDF values falling outside this range are invalid.
	8–15	COUNT	Count. Used only by the CP. Holds the number of cells sent since the last forward RM cell. Initialize with Nrm (in the parameter RAM).
0x16	—	ICR	Initial cell rate. The number of cells per second of the current ABR channel. The channel's ACR is initialized with ICR. ICR uses the ATMF TM 4.0 floating-point format.
0x18	—	CRM	Missing RM cells count. Limits the number of forward RM cells that may be sent in the absence of received backward RM cell. The CRM is in units of cells.
0x1A	—	ADTF	ADTF–ACR decrease time factor. The ADTF period is 500 ms as defined in the TM 4.0. The ADTF value is defined by the system clock and the time stamp timer prescaler; see Section 20.3.7, "RISC Time-Stamp Control Register (RTSCR)." For a time stamp prescaler of 1 µs, ADTF should be programmed to 500m/(1µs × 1024)= 488.

Table 35-27. ABR-Specific TCTE Field Descriptions (continued)

Offset	Bits	Name	Description
0x1C	—	ER	Explicit rate. Holds the explicit rate value (in cells/sec) of the current ABR channel. ER is copied to the F-RM cell ER field. The user usually initializes this field to PCR. ER uses the ATMF TM 4.0 floating-point format.
0x1E	—	ER-BRM	Explicit rate-backward RM cell. Holds the maximum explicit rate value (in cells/sec) allowed for B-RM cells. The ER-TA field which is inserted to each B-RM cell is limited by this value. ER-BRM uses the ATMF TM 4.0 floating-point format.

35.10.3 OAM Performance Monitoring Tables

The OAM performance monitoring tables include performance monitoring block test parameters, as shown in Figure 35-37. Each block test needs a 32-byte performance monitoring table in the dual-port RAM. In the connection's RCT and TCT, the user allocates an OAM performance table to a VCC or VPC. See Section 35.6.6, "Performance Monitoring." PMT_BASE in the parameter RAM points to the base address of the tables. The starting address of each PM table is given by PMT_BASE + RCT/TCT[PMT] × 32.

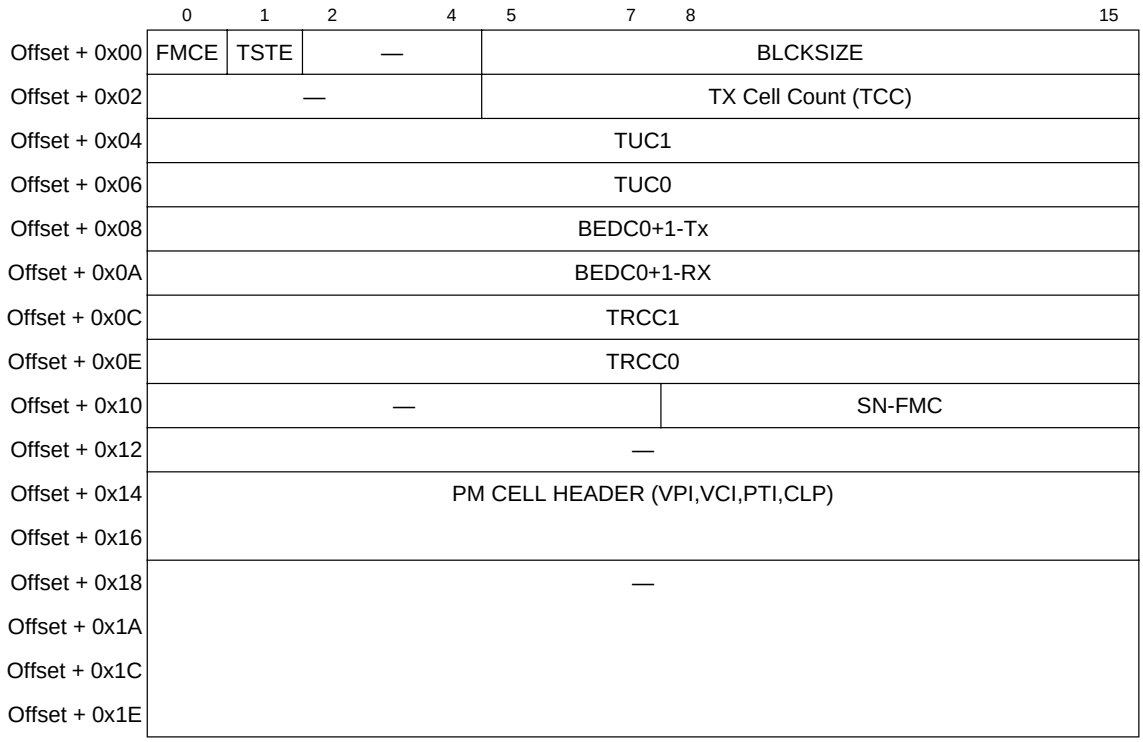


Figure 35-37. OAM Performance Monitoring Table

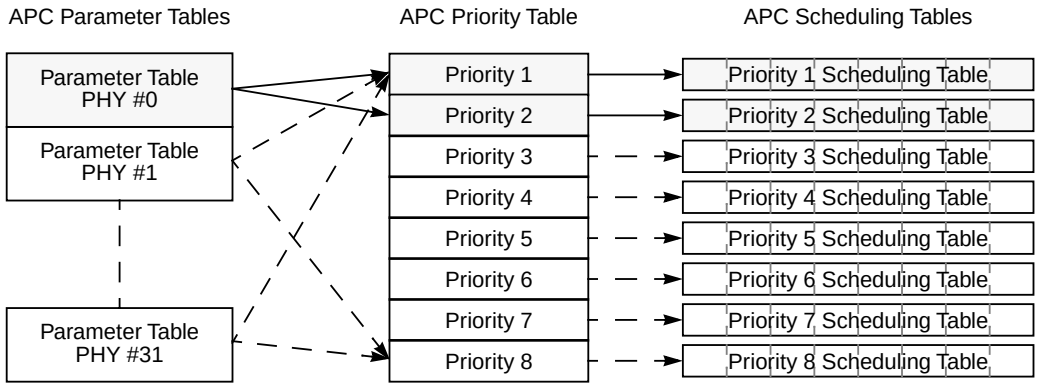
Table 35-28 describes fields in the performance monitoring table.

Table 35-28. . OAM—Performance Monitoring Table Field Descriptions

Offset	Bits	Name	Description
0x00	0	FMCE	Enables FMC transmission. Initialize to 1.
	1	TSTE	FMC time stamp enable 0 The time stamp field of the FMC is coded with all 1's. 1 The value of the time stamp timer is inserted into the time stamp field of the FMC.
	2–4	—	Reserved, should be cleared.
	5–15	BLCKSIZE	Performance monitoring block size ranging from 1 to 2,047 cells.
0x02	0–4	—	Reserved, should be cleared.
	5–15	TCC	TX cell count. Used by the CP to count data cells sent. Initialize to zero.
0x04	—	TUC1	Total user cell 1. Count of CLP = 1 user cells (modulo 65,536) sent. Should be cleared initially.
0x06	—	TUC0	Total user cell 0. Count of CLP = 0 user cells (modulo 65,536) sent. Should be cleared initially.
0x08	—	BEDC0+1-Tx	Block error detection code 0+1–transmitted cells. Even parity over the payload of the block of user cells sent since the last FMC. Should be cleared initially.
0x0A	—	BEDC0+1-RX	Block error detection code 0+1–received cells. Even parity over the payload of the block of user cells received since the last FMC. Should be cleared initially.
0x0C	—	TRCC1	Total received cell 1. Count of CLP = 1 user cells (modulo 65,536) received. Should be cleared initially.
0x0E	—	TRCC0	Total received cell 0. Count of CLP = 0 user cells (modulo 65,536) received. Should be cleared initially.
0x10	0–7	—	Reserved, should be cleared.
	8–15	SN-FMC	Sequence number of the last FMC sent. Should be cleared initially.
0x12	—	—	Reserved, should be cleared.
0x14	—	PMCH	PM cell header. Holds the ATM cell header of the FMC, BRC to be inserted by the CP into the Tx cell flow.
0x18–0x1E	—	—	Reserved, should be cleared.

35.10.4 APC Data Structure

The APC data structure consists of three elements: the APC parameter tables for the PHY devices, the APC priority table, and the APC scheduling tables. See Figure 35-38.



Note: The shaded areas represent the active structures for an example implementation of PHY #0 with two priorities. (The unshaded areas and dashed arrows represent unused structures.)

Figure 35-38. ATM Pace Control Data Structure

35.10.4.1 APC Parameter Tables

Each PHY's APC parameter table, shown in Table 35-29, holds parameters that define the priority table location, the number of priority levels, and other APC parameters. The table resides in the dual-port RAM. The parameter APCP_BASE, described in Section 35.10.1, "Parameter RAM," points to the base address of PHY#0's parameter table.

For multiple PHYs, the table structure is duplicated. Each table resides in 32 bytes of memory. The starting address of each APC parameter table is given by APCP_BASE + PHY# $\times$ 32. Note however that in slave mode with multiple PHYs, the parameter table always resides at APCP_BASE regardless of the PHY address.

Table 35-29. . APC Parameter Table

Offset [1]	Name	Width	Description
0x00	APCL_FIRST	Hword	Address of first entry in the priority table. Must be 8-byte aligned. User-initialized.
0x02	APCL_LAST	Hword	Address of last entry in the priority table. Must be 8-byte aligned. User-initialized as APCL_FIRST + 8 x (number_of_priorities - 1).
0x04	APCL_PTR	Hword	Address of current priority entry used by the CP. User-initialized with APCL_FIRST.
0x06	CPS	Byte	Cells per slot. Determines the number of cells sent per APC slot. See Section 35.3.2, "APC Unit Scheduling Mechanism." User-defined. (0x01 = 1 cell; 0xFF = 255 cells.) Note that if ABR is used, CPS must be a power of two.
0x07	CPS_CNT	Byte	Cells sent per APC slot counter. User-initialized to CPS; used by the CP.
0x08	MAX_ITERATION	Byte	Max iteration allowed. Number of scan iterations allowed in the APC. User-defined. This parameter limits the time spent in a single APC routine, thereby avoiding excessive APC latency.
0x09	CPS_ABR	Byte	ABR only. Cells per slot represented as a power of two. User-defined. (For example, if CPS is 1, CPS_ABR = 0x00; if CPS is 8, CPS_ABR = 0x03.)

Table 35-29. . APC Parameter Table (continued)

Offset [1]	Name	Width	Description
0x0A	LINE_RATE_ABR	Hword	ABR only. The PHY line rate in cells/sec, represented in TM 4.0 floating-point format. User-defined.
0xC	REAL_TSTP	Word	Real-time stamp pointer used internally by the APC. Should be cleared initially.
0x10	APC_STATE	Word	Used internally by the APC. Should be cleared initially.

[1]Offset values are to APCP_BASE+PHY# × 32. However, in slave mode, the offset is from APCP_BASE regardless of the PHY address.

35.10.4.2 APC Priority Table

Each PHY's APC priority table holds pointers to the APC scheduling table of each priority level. It resides in the dual-port RAM. The priority table can hold up to eight priority levels. Table 35-30 shows the structure of a priority table entry.

Table 35-30. APC Priority Table Entry

Offset	Name	Width	Description
0x00	APC_LEVi_BASE	Hword	APC level i base address. Pointer to the first slot in the APC scheduling table for level i. Should be half-word aligned. User-defined.
0x02	APC_LEVi_END	Hword	APC level i end address. Pointer to the last slot in the APC scheduling table for level i. Should be half-word aligned. User-defined.
0x04	APC_LEVi_RPTR	Hword	APC level i real-time/service pointers. APC table pointers used internally by the APC. Initialize both pointers to APC_LEVi_BASE.
0x06	APC_LEVi_SPTR	Hword	

35.10.4.3 APC Scheduling Tables

The APC uses APC scheduling tables (one table for each priority level) to schedule channel transmission. A scheduling table is divided into time slots, as shown in Figure 35-39. Each slot is a half-word entry. Note that the APC scheduling tables should be cleared before the APC unit is enabled.

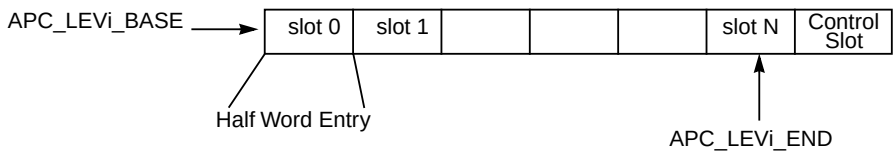


Figure 35-39. The APC Scheduling Table Structure

Slot N+1 is used as a control slot, as shown in Figure 35-40.

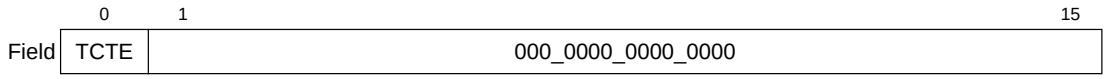


Figure 35-40. Control Slot

Table 35-31 describes the control slot.

Bits	Name	Description
0	TCTE	Used for external channels only. 0 Channels in this scheduling table do not use external TCTE. (No external VBR, ABR, UBR+ channels) 1 Channels in this scheduling table use external TCTE. (External VBR, ABR, UBR+ channels)
1–15	—	Reserved, should be cleared.

Table 35-31. Control Slot Field Description

35.10.5 ATM Controller Buffer Descriptors (BDs)

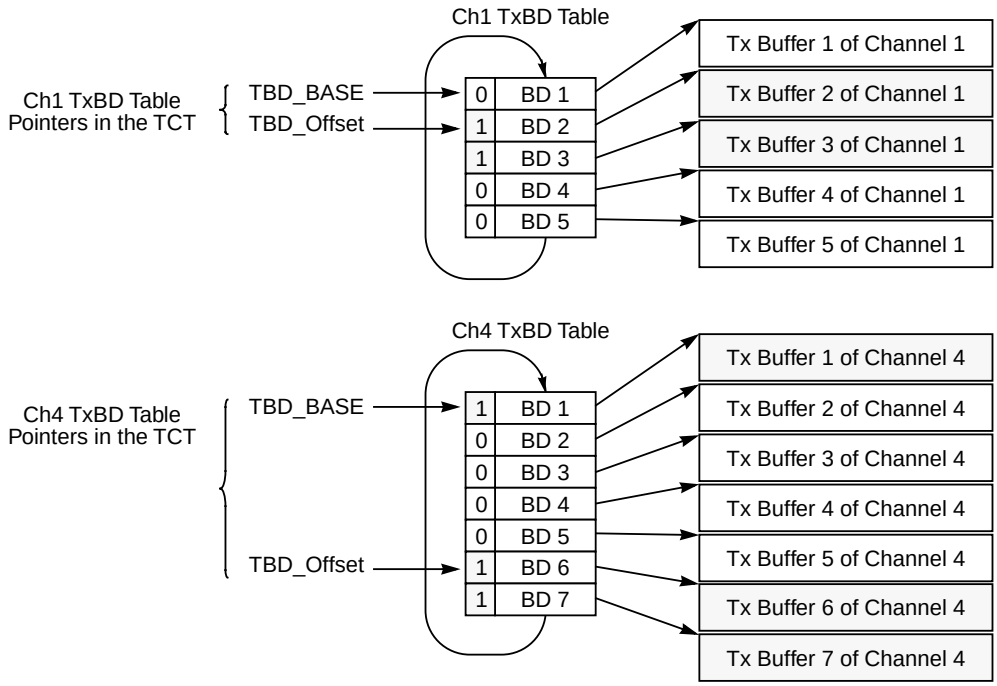
Each ATM channel has separate receive and transmit BD tables. The number of BDs per channel and the size of the buffers is user-defined. The last BD in each table holds a wrap indication. Each BD in the TxBD table points to a buffer to send. At the receive side, the user can choose one of two modes:

- Static buffer allocation. In this mode, the user allocates dedicated buffers to each ATM channel (that is, the user associates each BD with one buffer). Static buffer allocation is useful when the connection rate is known and constant and when data must be reassembled in a particular memory space.
- Global buffer allocation. Available for AAL5 only. In this mode, buffer allocation is dynamic. The user allocates receive buffers and places them in global buffer pools. When the CP needs a receive buffer, it first fetches a buffer pointer from one of the global buffer pools and writes the pointer to the current RxBD. Global buffer allocation is optimized for allocating memory among many ATM channels with variable data rates, such as ABR channels.

35.10.5.1 Transmit Buffer Operation

The user prepares a table of BDs pointing to the buffers to be sent. The address of the first BD is put in the channel's TCT[TBD_BASE]. The transmit process starts when the core issues an ATM TRANSMIT command. The CP reads the first TxBD in the table and sends its associated buffer. When the current buffer is finished, the CP increments TBD_Offset, which holds the offset from TBD_BASE to the current BD. It then reads the next BD in the table. If the BD is ready (TxBD[R] = 1), the CP continues sending. If the current BD is not ready, the CP polls the ready bit at the channel rate unless TCT[AVCF] = 1, in which case the CP removes the channel from the APC and clears TCT[VCON]. The core must issue a new ATM TRANSMIT command to restart transmission.

Figure 35-41 shows the ready bit in the TxBD tables and their associated buffers for two example ATM channels.



Note: The shaded buffers are ready to be sent; unshaded buffers are waiting to be prepared.

Figure 35-41. Transmit Buffers and BD Table Example

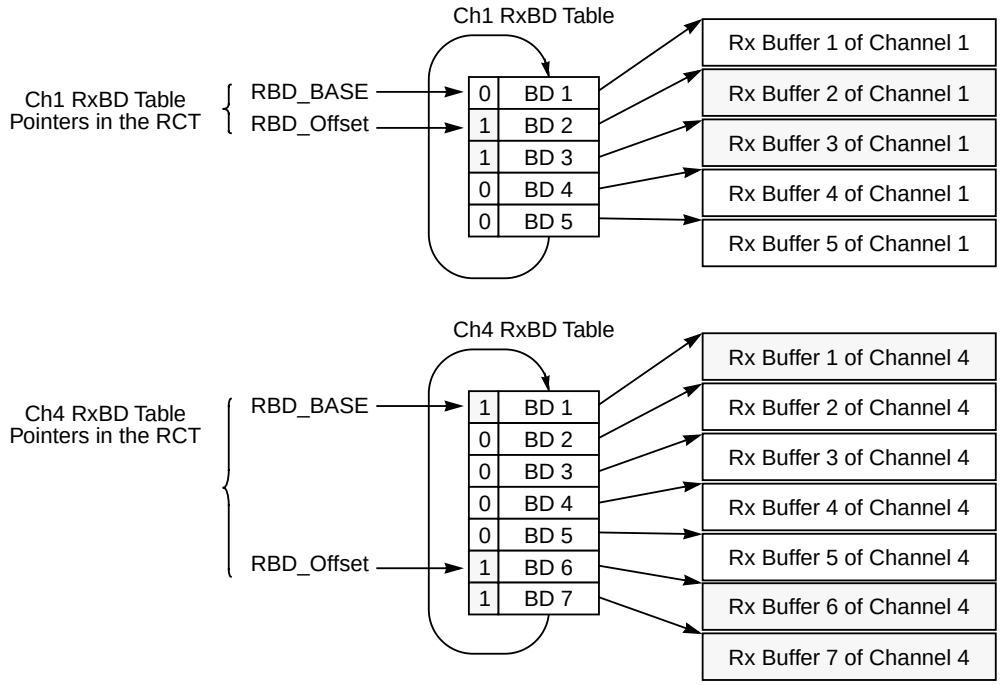
35.10.5.2 Receive Buffer Operation

For AAL5 channels, the user should choose to operate in static buffer allocation or in global buffer allocation by writing to RCT[BUFM]. AAL1 and AAL0 channels must use static buffer allocation.

35.10.5.2.1 Static Buffer Allocation

The user prepares a table of BDs pointing to the receive buffers. The address of the first BD is put in the channel's RCT[RBD_BASE]. When an ATM cell arrives, the CP opens the first BD in the table and starts filling its associated buffer with received data. When the current buffer is full, the CP increments RBD_Offset, which is the offset to the current BD from RBD_BASE, and reads the next BD in the table. If the BD is empty (RxBD[E] = 1), the CP continues receiving. If the BD is not empty, a busy condition has occurred and a busy interrupt is sent to the event queue.

Figure 35-42 shows the empty bit in the RxBD tables and their associated buffers for two example ATM channels.



Note: The shaded buffers are empty; unshaded buffers are waiting to be processed.

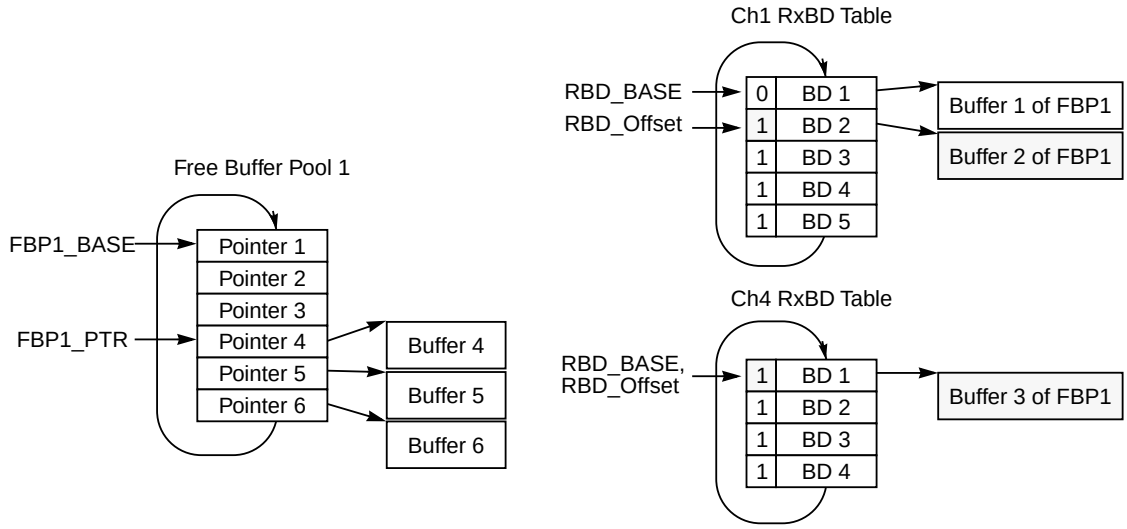
Figure 35-42. Receive Static Buffer Allocation Example

35.10.5.2.2 Global Buffer Allocation

The user prepares a table of BDs without assigning buffers to them (no buffer pointers). The address of the first BD is put into the channel's RCT[RBD_BASE]. The user also prepares sets of free buffers (of size RCT[MRBLR]) in up to four free buffer pools (chosen in RCT[BPOOL]); see Section 35.10.5.2.3, "Free Buffer Pools."

When an ATM cell arrives, the CP opens the first BD in the table, fetches a buffer pointer from the free buffer pool associated with this channel, and writes the pointer to RxBD[RXDBPTR], the receive data buffer pointer field in the BD. When the current buffer is full, the CP increments RBD_Offset, which is the offset from the RBD_BASE to the current BD, and reads the next BD in the table. If the BD is empty (RxBD[E] = 1), the CP fetches another buffer pointer from the free buffer pool and reception continues. If the BD is not empty, a busy condition occurs and a busy interrupt is sent to the event queue specifying the ATM channel code. As software then processes each full buffer (RxBD[E] = 0), it sets RxBD[E] and copies the buffer pointer back to the free buffer pool.

Figure 35-43 shows two ATM channels' BD tables and one free buffer pool. Both channels are associated with free buffer pool 1. The CP allocates the first two buffers of buffer pool 1 to channel 1 and the third to channel 4.



Notes: Buffers 2 and 3 are receiving data. After buffer 1 is processed, it can be returned to the pool.

Figure 35-43. Receive Global Buffer Allocation Example

35.10.5.2.3 Free Buffer Pools

As Figure 35-44 shows, when a buffer pointer is fetched from a pool, the CP clears the entry's valid bit and increments FBP#_PTR. After the CP uses an entry with the wrap bit set (W = 1), it returns to the first entry in the pool. After a buffer pointer is returned to the pool, the user should set V to indicate that the entry is valid. If the CP tries to read an invalid entry (V = 0), the buffer pool is out of free buffers; the global-buffer-pool-busy event is then set in FCCE[GBPB] and a busy interrupt is sent to the interrupt queue specifying the ATM channel code associated with the pool.

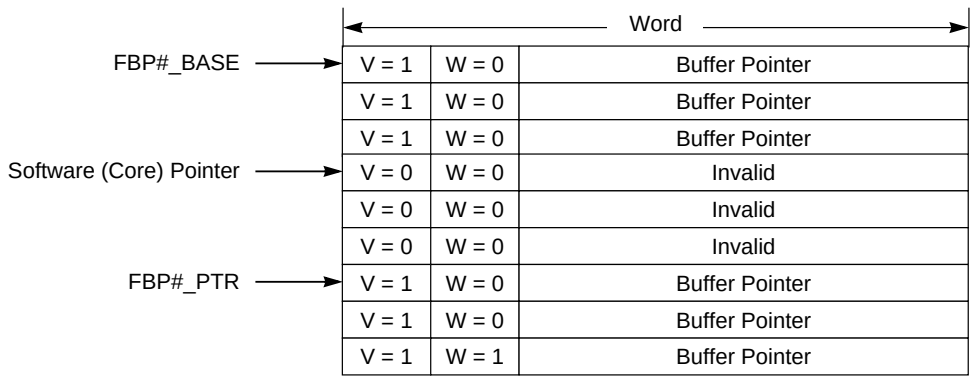


Figure 35-44. Free Buffer Pool Structure

Figure 35-45 describes the structure of a free buffer pool entry.

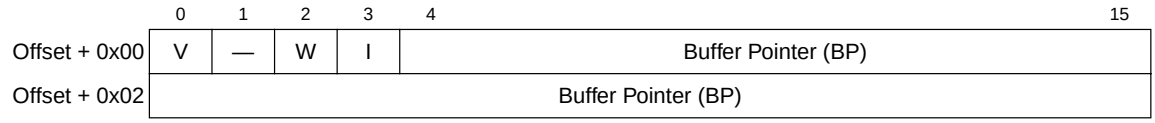


Figure 35-45. Free Buffer Pool Entry

Table 35-32 describes free buffer pool entry fields.

Table 35-32. Free Buffer Pool Entry Field Descriptions

Offset	Bits	Name	Description
0x00	0	V	Valid buffer entry. 0 This free buffer pool entry contains an invalid buffer pointer. 1 This free buffer pool entry contains a valid buffer pointer.
	1	—	Reserved, should be cleared.
	2	W	Wrap bit. When set, this bit indicates the last entry in the circular table. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3	I	Red-line interrupt. Can be used to indicate that the free buffer pool has reached a red line and additional buffers should be added to this pool to avoid a busy condition. 0 No interrupt is generated. 1 A red-line interrupt is generated when this buffer is fetched from the free buffer pool.
	4–15	BP	Buffer pointer. Points to the start address of the receive buffer. The four msbs are control bits, and the four msbs of the real buffer pointer are taken from the four msbs of the parameter FBP_ENTRY_EXT in the free buffer pool parameter table.
0x02	0–15		

35.10.5.2.4 Free Buffer Pool Parameter Tables

The free buffer pool parameters are held in parameter tables in the dual-port RAM; see Table 35-33. FBT_BASE in the parameter RAM points to the base address of these tables. Each of the four free buffer pools has its own parameter table with a starting address given by FBT_BASE+ RCT[BPOOL] $\times$ 16.

Table 35-33. Free Buffer Pool Parameter Table

Offset [1]	Bits	Name	Description
0x00	—	FBP_BASE	Free buffer pool base. Holds the pointer to the first entry in the free buffer pool. FBP_BASE should be word aligned. User-defined.
0x04	—	FBP_PTR	Free buffer pool pointer. Pointer to the current entry in the free buffer pool. Initialize to FBP_BASE.
0x08	—	FBP_ENTRY_EXT	Free buffer pool entry extension. FBP_ENTRY_EXT[0–3] holds the four left bits of FBP_ENTRY. FBP_ENTRY_EXT[4–15] should be cleared. User-defined.
0x0A	0	BUSY	The CP sets this bit when it tries to fetch buffer pointer with V bit clear. FCCE[GBPB] is also set. Initialize to zero.
	1	RLI	Red-line interrupt. Set by the CP when it fetches a buffer pointer with I = 1. FCCE[GRLI] is also set. Initialize to zero.
	2–7	—	Reserved, should be cleared.

Table 35-33. Free Buffer Pool Parameter Table (continued)

Offset [1]	Bits	Name	Description
	8	EPD	Early packet discard. 0 Normal operation. 1 AAL5 frames in progress are received, but new AAL5 frames associated with this pool are discarded. Can be used to implement EPD under core control.
	9–15	—	Reserved, should be cleared.
0x0C	—	FBP_ENTRY	Free buffer pool entry. Initialize with the first entry of the free buffer pool. Note that FBP_ENTRY must be reinitialized with the entry pointed to by FBP_PTR when a busy state occurs to reenable free buffer pool processing.

[1] Offset from FBT_BASE+RCT[BPOOL] × 16

35.10.5.3 ATM Controller Buffers

Table 35-34 describes properties of the ATM receive and transmit buffers.

Table 35-34. Receive and Transmit Buffers

AAL	Receive		Transmit	
	Size	Alignment	Size	Alignment
AAL5	Multiple of 48 octets (except last buffer in frame)	Double word aligned	Any	No requirement
AAL1	At least 47 octets	No requirement	At least 47 octets	No requirement
AAL0	52-64 octets.	Burst-aligned	52–64 octets.	No requirement

35.10.5.4 AAL5 RxBD

Figure 35-46 shows the AAL5 RxBD.

	0	1	2	3	4	5	6	7–9	10	11	12	13	14	15
Offset + 0x00	E	—	W	I	L	F	CM	—	CLP	CNG	ABRT	CPUU	LNE	CRE
Offset + 0x02	Data Length (DL)													
Offset + 0x04	Rx Data Buffer Pointer (RXDBPTR)													
Offset + 0x06														

Figure 35-46. AAL5 RxBD

Table 35-35 describes AAL5 RxBD fields.

Table 35-35. AAL5 RxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	E	Empty. 0 The buffer associated with this RxBD is full or data reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP does not use this BD again while E remains zero. 1 The buffer associated with this RxBD is empty or reception is in progress. This RxBD and its receive buffer are controlled by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	—	Reserved, should be cleared.
	2	W	Wrap (final BD in table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the CP receives incoming data into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	I	Interrupt 0 No interrupt is generated after this buffer has been used. 1 An Rx buffer event is sent to the interrupt queue after the ATM controller uses this buffer. FCCE[GINT*x*] is set in the event register when INT_CNT reaches the global interrupt threshold.
	4	L	Last in frame. Set by the ATM controller for the last buffer in a frame. 0 Buffer is not last in a frame. 1 Buffer is last in a frame. ATM controller writes frame length in DL and updates the error flags.
	5	F	First in frame. Set by the ATM controller for the first buffer in a frame. 0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
	6	CM	• Continuous mode 0 Normal operation. 1 The CP does not clear the empty bit after this BD is closed, allowing the associated buffer to be overwritten automatically when the CP next accesses this BD.
	7–9	—	Reserved, should be cleared.
	10	CLP	Cell loss priority. At least one cell associated with the current message was received with CLP = 1. May be set at the last buffer of the message.
	11	CNG	Congestion indication. The last cell associated with the current message was received with PTI middle bit set. CNG may be set at the last buffer of the message.
	12	ABRT	Abort message indication. The current message was received with Length field zero.
	13	CPUU	CPCS-UU+CPI indication. Set when the CPCS-UU+CPI field is non zero. CPUU may be set at the last buffer of the message.
	14	LNE	Rx length error. AAL5 CPCS-PDU length violation. May be set only for the last BD of the frame if the pad length is greater than 47 or less than zero octets.
	15	CRE	Rx CRC error. Indicates CRC32 error in the current AAL5 PDU. Set only for the last BD of the frame.

Table 35-35. AAL5 RxBD Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	—	DL	Data length. The number of octets written by the CP into this BD's buffer. It is written by the CP once the BD is closed. In the last BD of a frame, DL contains the total frame length.
0x04		RXDBPTR	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in internal or external memory. This pointer must be burst-aligned.

35.10.5.5 AAL1 RxBD

Figure 35-47 shows the AAL1 RxBD.

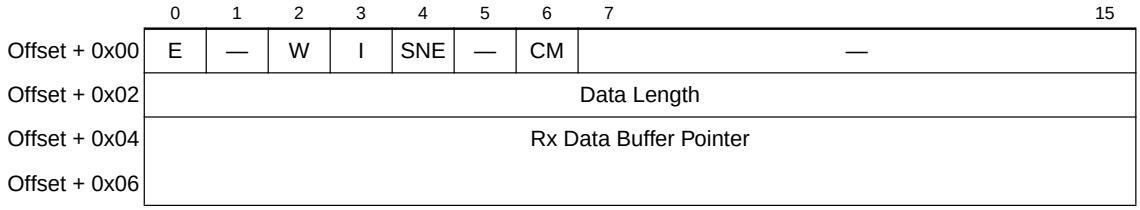


Figure 35-47. AAL1 RxBD

Table 35-36 describes AAL1 RxBD fields.

Table 35-36. AAL1 RxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	E	Empty 0 The buffer associated with this RxBD is filled with received data or data reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP cannot use this BD again while E = 0. 1 The buffer is not full. This RxBD and its associated receive buffer are owned by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	—	Reserved, should be cleared.
	2	W	Wrap (final BD in table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of this current channel. After this buffer is used, the CP receives incoming data into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table overall space is constrained to 64 Kbytes.
	3	I	Interrupt 0 No interrupt is generated after this buffer has been used. 1 An Rx buffer event is sent to the interrupt queue after the ATM controller uses this buffer. FCCE[GINT*x*] is set when the INT_CNT reaches the global interrupt threshold.
	4	SNE	Sequence number error. SNE is set when a sequence number error is detected in the current AAL1 buffer.
	5	—	Reserved, should be cleared.
	6	CM	Continuous mode 0 Normal operation. 1 The empty bit (RxBD[E]) is not cleared by the CP after this BD is closed, allowing the associated buffer to be overwritten automatically when the CP next accesses this BD.
	7–15	—	Reserved, should be cleared.
0x02	—	DL	Data length. The number of octets the CP writes into the buffer once its BD is closed.
0x04	—	RXDBPTR	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in either internal or external memory. This pointer must be burst-aligned.

35.10.5.6 AAL0 RxBD

Figure 35-48 shows the AAL0 RxBD.

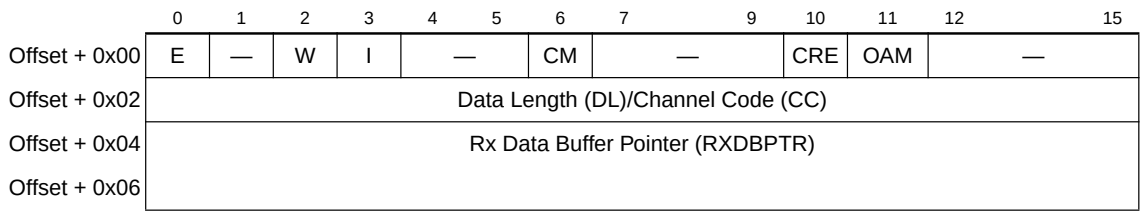


Figure 35-48. AAL0 RxBD

Table 35-37 describes AAL0 RxBD fields.

Table 35-37. AAL0 RxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	E	Empty 0 The buffer associated with this RxBD is filled with received data, or data reception was aborted due to an error. The core can examine or write to any fields of this RxBD. The CP does not use this BD again while E remains zero. 1 The Rx buffer is empty or reception is in progress. This RxBD and its associated receive buffer are owned by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	—	Reserved, should be cleared.
	2	W	Wrap (final BD in table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of the current channel. After this buffer has been used, the CP will receive incoming data into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	I	Interrupt 0 No interrupt is generated after this buffer has been used. 1 An Rx buffer event is sent to the interrupt queue after the ATM controller uses this buffer. FCCE[GINT*x*] is set when the INT_CNT reaches the global interrupt threshold.
	4–5	—	Reserved, should be cleared.
	6	CM	Continuous mode 0 Normal operation. 1 The CP does not clear the E bit after this BD is closed, allowing the associated buffer to be overwritten automatically when the CP next accesses this BD.
	7–9	—	Reserved, should be cleared.
	10	CRE	Rx CRC error. Indicates a CRC10 error in the current AAL0 buffer. The CRE bit is considered an error only if the received cell had a CRC10 field in the cell payload.
	11	OAM	Operation and maintenance cell. If OAM is set, the current AAL0 buffer contains an OAM cell. This cell is associated with the channel indicated by the channel code field (CC field).
	12-15	—	Reserved, should be cleared.
0x02	—	DL/CC	Data length/channel code. If RxBD[OAM] is set, this field functions as CC; otherwise, it is DL. Data length is the size in octets of this buffer (MRBLR value). Channel code specifies the channel code associated with this OAM cell.
0x04	—	RXDBPTR	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in either internal or external memory. This pointer must be burst-aligned.

35.10.5.7 AAL5, AAL1 User-Defined Cell—RxBD Extension

In user-defined cell mode, the AAL5 and AAL1 RxBDs are extended to 32 bytes; see Figure 35-49. Note that for AAL0, a complete cell, including the UDC header, is stored in the buffer; the AAL0 BD size is always 8 bytes.

Offset + 0x08	Extra Cell Header. Used to store the user-defined cell's extra cell header. The extra cell header can be 1–12 bytes long.
Offset + 0x14	Reserved (12 bytes)

Figure 35-49. User-Defined Cell—RxBD Extension

35.10.5.8 AAL5 TxBDs

Figure 35-50 shows the AAL5 TxBD.

	0	1	2	3	4	5	6	7 9	10	11	12 15
Offset + 0x00	R	—	W	I	L	—	CM	—	CLP	CNG	—
Offset + 0x02	Data Length (DL)										
Offset + 0x04	Tx Data Buffer Pointer (TXDBPTR)										
Offset + 0x06											

Figure 35-50. AAL5 TxBD

Table 35-38 describes AAL5 TxBD fields.

Table 35-38. AAL5 TxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	R	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The CP clears R after the buffer is sent or after an error condition is encountered. 1 The user-prepared buffer has not been sent or is currently being sent. No fields of this BD may be written by the user once R is set.
	1	—	Reserved, should be cleared.
	2	W	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer is used, the CP sends outgoing data from the first BD in the table (the BD pointed to by the channel's TCT[TBD_BASE]). The number of TxBDs in this table is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	I	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx Buffer event is sent to the interrupt queue after this buffer is serviced. FCCE[GINT*x*] is set when the INT_CNT counter reaches the global interrupt threshold.
	4	L	Last in frame. Set by the user to indicate the last buffer in a frame. 0 Buffer is not last in a frame. 1 Buffer is last in a frame.
	5	—	Reserved, should be cleared.
	6	CM	Continuous mode 0 Normal operation. 1 The CP does not clear R after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of CM.
	7-9	—	Reserved, should be cleared.
	10	CLP	The ATM cell header CLP bit of the cells associated with the current frame are ORed with this field. This field is valid only in the first BD of the frame.
	11	CNG	The ATM cell header CNG bit of the cells associated with the current frame are ORed with this field. This field is valid only in the first BD of the frame.
	12–15	—	Reserved, should be cleared.
0x02	—	DL	The number of octets the ATM controller should transmit from this BD's buffer. It is not modified by the CP. The value of DL should be greater than zero.
0x04	—	TXDBPTR	Tx data buffer pointer. Points to the address of the associated buffer, which may or may not be 8-byte-aligned. The buffer may reside in either internal or external memory. This value is not modified by the CP.

35.10.5.9 AAL1 TxBDs

Figure 35-51 shows the AAL1 TxBD.

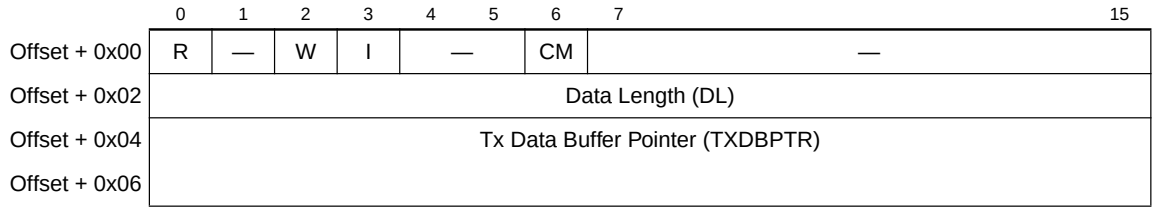


Figure 35-51. AAL1 TxBD

Table 35-39 describes AAL1 TxBD fields.

Table 35-39. AAL1 TxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	R	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The CP clears this bit after the buffer has been sent or after an error condition is encountered. 1 The buffer prepared for transmission by the user has not been sent or is being sent. No fields of this BD may be written by the user once R is set.
	1	—	Reserved, should be cleared.
	2	W	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer is used, the CP sends outgoing data from the first BD in the table (the BD pointed to by the channel's TCT[TBD_BASE]). The number of TxBDs in this table is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	I	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx buffer event is sent to the interrupt queue after this buffer is serviced. FCCE[GINT*x*] is set when the INT_CNT counter reaches the global interrupt threshold.
	4–5	—	Reserved, should be cleared.
	6	CM	Continuous mode 0 Normal operation. 1 The CP does not clear the ready bit after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD.
	7–11	—	Reserved, should be cleared.
0x02	—	DL	The number of octets the ATM controller should transmit from this BD's buffer. It is not modified by the CP. The value of DL should be greater than zero.
0x04	—	TXDBPTR	Tx data buffer pointer. Points to the address of the associated buffer. The buffer may reside in either internal or external memory. This value is not modified by the CP.

35.10.5.10 AAL0 TxBDs

Figure 35-52 shows AAL0 TxBDs. Note that the data length field is calculated internally as 52 bytes, plus the extra header length (defined in FPSMR[TEHS]) when in UDC mode.

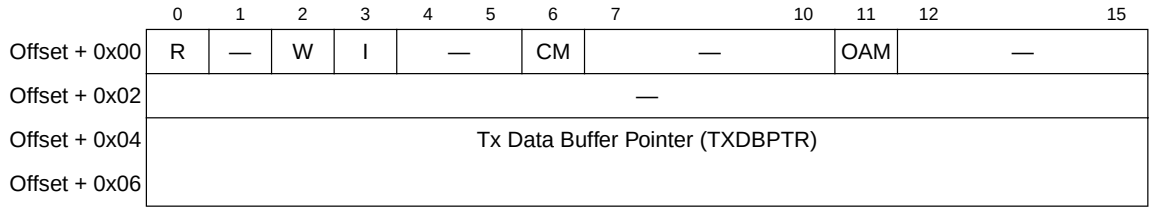


Figure 35-52. AAL0 TxBDs

Table 35-40 describes AAL0 TxBD fields.

Table 35-40. AAL0 TxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	R	Ready 0 The buffer is not ready for transmission. The user can manipulate this BD or its buffer. The CP clears R after the buffer has been sent or after an error occurs. 1 The buffer that the user prepared for transmission has not been sent or is being sent. No fields of this BD may be written by the user once R is set.
	1	—	Reserved, should be cleared.
	2	W	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer is used, the CP sends outgoing data from the first BD in the table (the BD pointed to by the channel's TCT[TBD_BASE]). The number of TxBDs in this table is determined by the W bit. The current table is constrained to 64 Kbytes.
	3	I	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx buffer event is sent to the interrupt queue after this buffer is serviced. FCCE[GINT*x*] is set when the INT_CNT counter reaches the global interrupt threshold.
	4–5	—	Reserved, should be cleared.
	6	CM	Continuous mode 0 Normal operation. 1 The CP does not clear the ready bit after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD.
	7–10	—	Reserved, should be cleared.
	11	OAM	Operation and maintenance cell. If OAM is set, the current AAL0 buffer contains an F5 or F4 OAM cell. Performance monitoring calculations are not done on OAM cells.
	11–15	—	Reserved, should be cleared.
0x02	—	—	Reserved, should be cleared.
0x04	—	TXDBPTR	Tx data buffer pointer. Points to the address of the associated buffer, which may or may not be 8-byte-aligned. The buffer may reside in either internal or external memory. This value is not modified by the CP.

35.10.5.11 AAL5, AAL1 User-Defined Cell—TxBD Extension

In user-defined cell mode, the AAL5 and AAL1 TxBDs are extended to 32 bytes; see Figure 35-53. Note that for AAL0 a complete cell, including the UDC header, is stored in the buffer; the AAL0 BD size is always 8 bytes.

Offset + 0x08	Extra Cell Header. Used to store the user-defined cell's extra cell header. The extra cell header can be 1–12 bytes long.
Offset + 0x14	Reserved (12 bytes)

Figure 35-53. User-Defined Cell—TxBD Extension

35.10.6 AAL1 Sequence Number (SN) Protection Table (AAL1 Only)

The 32-byte sequence number protection table, pointed to by AAL1_SNPT_BASE in the ATM parameter RAM, resides in dual-port RAM and is used for AAL1 only. The table should be initialized according to Figure 35-54.

	0 15
Offset + 0x00	0x0000
Offset + 0x02	0x0007
Offset + 0x04	0x000D
Offset + 0x06	0x000A
Offset + 0x08	0x000E
Offset + 0x0A	0x0009
Offset + 0x0C	0x0003
Offset + 0x0E	0x0004
Offset + 0x10	0x000B
Offset + 0x12	0x000C
Offset + 0x14	0x0006
Offset + 0x16	0x0001
Offset + 0x18	0x0005
Offset + 0x1A	0x0002
Offset + 0x1C	0x0008
Offset + 0x1E	0x000F

Figure 35-54. AAL1 Sequence Number (SN) Protection Table

35.10.7 UNI Statistics Table

The UNI statistics table, shown in Table 35-41, resides in the dual-port RAM and holds UNI statistics parameters. UNI_STATT_BASE points to the base address of this table. Each PHY has its own table with a starting address given by UNI_STATT_BASE+ PHY# × 8.

Table 35-41. UNI Statistics Table

Offset [1]	Name	Width	Description
0x00	UTOPIAE	Hword	Counts cells dropped as a result of UTOPIA parity error or state machine errors (short or long cells).
0x02	MIC_COUNT	Hword	Counts misinserted cells dropped as a result of address look-up failure.
0x04	CRC10E_COUNT	Hword	Counts cells dropped as a result of CRC10 failure. AAL5-ABR only.
0x06	—	Hword	Reserved, should be cleared.

[1] Offset from UNI_STATT_BASE+PHY# × 8

35.11 ATM Exceptions

The ATM controller interrupt handling involves two principal data structures: FCCEs (FCC event registers) and circular interrupt queues.

Four priority interrupt queues are available. By programming RCT[INTQ] and TCT[INTQ], the user determines which queue receives the interrupt. Channel Rx buffer, Rx frame, or Tx buffer events can be masked by clearing interrupt mask bits in RCT and TCT.

After an interrupt request, the host reads FCCE. If FCCE[GINT*x*] = 1, at least one entry was added to one of the interrupt queues. After clearing FCCE[GINT*x*], the host processes the valid interrupt queue entries and clears each entry's valid bit. The host follows this procedure until it reaches an entry with V = 0. See Section 35.11.2, "Interrupt Queue Entry."

The host controls the number of interrupts sent to the core using a down counter in the interrupt queue's parameter table; see Section 35.11.3. For each event sent to an interrupt queue, a counter (that has been initialized to a threshold number of interrupts) is decremented. When the counter reaches zero, the global interrupt, FCCE[GINT*x*], is set.

35.11.1 Interrupt Queues

Interrupt queues are located in external memory. The parameters of each queue are stored in a table. See Section 35.11.3, "Interrupt Queue Parameter Tables."

When an interrupt occurs, the CP writes a new entry to the interrupt queue, the V bit is set, and the queue pointer (INTQ_PTR) is incremented. Once the CP uses an entry with W = 1,

it returns to the first entry in the queue. If the CP tries to overwrite a valid entry (V = 1), an overflow condition occurs and the queue's overflow flag, FCCE[INTO*x*], is set.

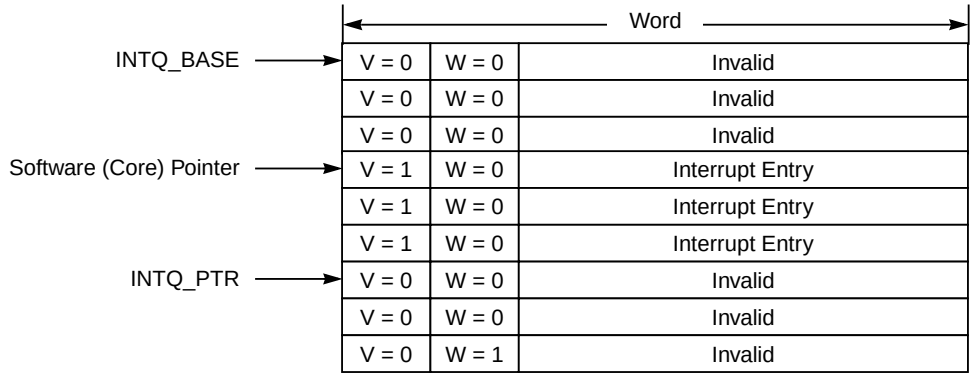


Figure 35-55. Interrupt Queue Structure

35.11.2 Interrupt Queue Entry

Each one-word interrupt queue entry provides detailed interrupt information to the host. Figure 35-56 shows an entry.

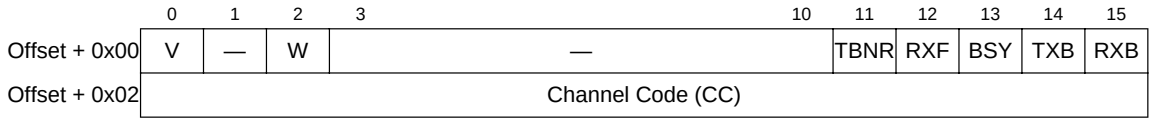


Figure 35-56. Interrupt Queue Entry

Table 35-42 describes interrupt queue entry fields.

Table 35-42. Interrupt Queue Entry Field Description

Offset	Bits	Name	Description
0x00	0	V	Valid interrupt entry 0 This interrupt queue entry is free and can be use by the CP. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	—	Reserved, should be cleared.
	2	W	Wrap bit. When set, this is the last interrupt circular table entry. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	—	Reserved, should be cleared.
	11	TBNR	Tx buffer-not-ready. Set when a transmit buffer-not-ready interrupt is issued. This interrupt is issued when the CP tries to open a TxBD that is not ready (R = 0). This interrupt is sent only if TCT[BNM] = 1. This interrupt has an associated channel code. Note that for AAL5, this interrupt is sent only if frame transmission is started. In this case, an abort frame transmission is sent (last cell with length=0), the channel is taken out of the APC, and the TCT[VCON] flag is cleared.
	12	RXF	Rx frame. RXF is set when an Rx frame interrupt is issued. This interrupt is issued at the end of AAL5 PDU reception. This interrupt is issued only if RCT[RXFM] = 1. This interrupt has an associated channel code.
	13	BSY	Busy condition. The BD table or the free buffer pool associated with this channel is busy. Cells were discarded due to this condition. This interrupt has an associated channel code.
	14	TXB	Tx buffer. TXB is set when a transmit buffer interrupt is issued. This interrupt is enabled when both TxBD[I] and TCT[IMK] = 1. This interrupt has an associated channel code.
	15	RXB	Rx buffer. RXB is set when an Rx buffer interrupt is issued. This interrupt is enabled when both RxBD[I] and RCT[RXBM] = 1. This interrupt has an associated channel code.
0x02	—	CC	Channel code specifies the channel associated with this interrupt.

35.11.3 Interrupt Queue Parameter Tables

The interrupt queue parameters are held in parameter tables in the dual-port RAM; see Table 35-43. INTT_BASE in the parameter RAM points to the base address of these tables. Each of the four interrupt queues has its own parameter table with a starting address given by INTT_BASE+ RCT/TCT[INTQ] × 16.

Table 35-43. Interrupt Queue Parameter Table

Offset [1]	Name	Width	Description
0x00	INTQ_BASE	Word	Base address of the interrupt queue. User-defined.
0x04	INTQ_PTR	Word	Pointer to interrupt queue entry. Initialize to INTQ_BASE.
0x08	INT_CNT	Half Word	Interrupt counter. Initialize with INT_ICNT. The CP decrements INT_CNT for each interrupt. When INT_CNT reaches zero, the queue's global interrupt flag FCCE[GINT*x*] is set.

Table 35-43. Interrupt Queue Parameter Table (continued)

Offset [1]	Name	Width	Description
0x0A	INT_ICNT	Half Word	Interrupt initial count. User-defined global interrupt threshold—the number of interrupts required before the CP issues a global interrupt (FCCE[GINT*x*]).
0x0C	INTQ_ENTRY	Word	Interrupt queue entry. Must be initialized to the entry pointed to by INTQ_PTR, which is initially the first empty entry of the queue. Note that after an overrun occurs, this entry must be reset to the entry pointed to by INTQ_PTR to reenable interrupt processing.

[1] Offset from INTT_BASE+RCT/TCT[INTQ] × 16

35.12 The UTOPIA Interface

The ATM controller interfaces with a PHY device through the UTOPIA interface. The MPC8560 supports UTOPIA level 2 for both master and slave modes.

35.12.1 UTOPIA Interface Master Mode

UTOPIA master signals are shown in Figure 35-57.

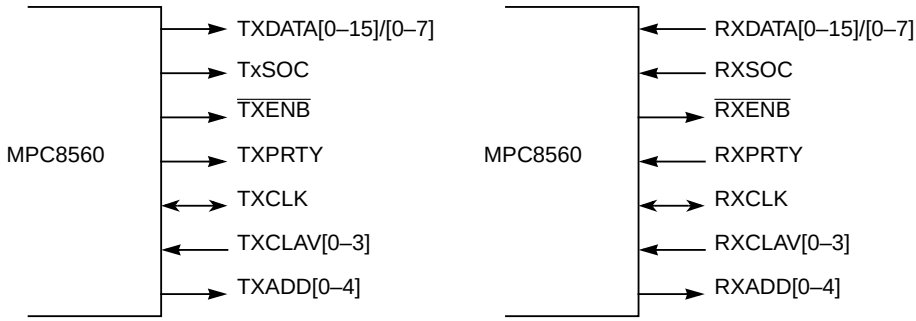


Figure 35-57. UTOPIA Master Mode Signals

Table 35-44 describes UTOPIA master mode signals.

Table 35-44. UTOPIA Master Mode Signal Descriptions

Signal	Description
TxDATA[0–15]/[0–7]	Carries transmit data from the ATM controller to a PHY device. TxDATA[15]/[7] is the msb when using UTOPIA 16/8, TxDATA[0] is the lsb.
TxSOC	Transmit start of cell. Asserted by the ATM controller when the first byte of a cell is sent on TxDATA lines.
$\overline{\text{TxENB}}$	Transmit enable. Asserted by the ATM controller when valid data is placed on the TxDATA lines.
TxCLAV[0–3]	Transmit cell available. Asserted by the PHY device to indicate that the PHY has room for a complete cell.
TxPRTY	Transmit parity. Asserted by the ATM controller. It is an odd parity bit over the TxDATA bits.

Table 35-44. UTOPIA Master Mode Signal Descriptions (continued)

Signal	Description
TxCLK	Transmit clock. Provides the synchronization reference for the TxDATA, TxSOC, $\overline{\text{TxENB}}$, TxCLAV, TxPRTY signals. All the above signals are sampled at low-to-high transitions of TxCLK.
TxADD[0–4]	Transmit address. Address bus from the ATM controller to the PHY device used to select the appropriate M-PHY device. Each M-PHY device needs to maintain its address. TxADD[4] is the msb.
RxDATA[0–15] /[0–7]	Carries receive data from the PHY to the ATM controller. RxDATA[15]/[7] is the msb when using UTOPIA 16/8, RxDATA[0] is the lsb.
RxSOC	Receive start of cell. Asserted by the PHY device as the first byte of a cell is received on RxDATA.
RxENB	Receive enable. An ATM controller asserts to indicate that RxDATA and RxSOC will be sampled at the end of the next RxCLK cycle. For multiple PHYs, $\overline{\text{RxENB}}$ is used to three-state RxDATA and RxSOC at each PHY's output. RxDATA and RxSOC should be enabled only in cycles after those with $\overline{\text{RxENB}}$ asserted.
RxCLAV[0–3]	Receive cell available. Asserted by a PHY device when it has a complete cell to give the ATM controller.
RxPRTY	Receive parity. Asserted by the PHY device. It is an odd parity bit over the RxDATA. If there is a RxPRTY error and the receive parity check FPSMR[RxP] is enabled, the cell is discarded. See Section 35.13.2, "FCC Protocol-Specific Mode Register (FPSMR)."
RxCLK	Receiver clock. Synchronization reference for RxDATA, RxSOC, $\overline{\text{RxENB}}$, RxCLAV, and RxPRTY, all of which are sampled at low-to-high transitions of RxCLK.
RxADD[0–4]	Receive address. Address bus from the ATM controller to the PHY device used to select the appropriate M-PHY device. Each M-PHY device needs to maintain its address. RxADD[4] is the msb.

35.12.1.1 UTOPIA Master Multiple PHY Operation

The cell transfer in a multiple PHY ATM port uses cell-level handshaking as defined in the UTOPIA standards. The MPC8560 supports two polling modes:

- Direct polling uses CLAV[0–3] with PHY selection using ADD[0–2]. Up to four PHYs can be supported.
- Multiplex polling uses CLAV[0] and ADD[0–4]. ATM controller polls all active PHYs starting from PHY address 0x0 to the address written in FPSMR[LAST_PHY]. Up to 31 PHY devices are supported.

Both modes support round-robin priority or fixed priority, described in Section 35.13.2, "FCC Protocol-Specific Mode Register (FPSMR)."

35.12.2 UTOPIA Interface Slave Mode

UTOPIA slave signals are shown in Figure 35-58.

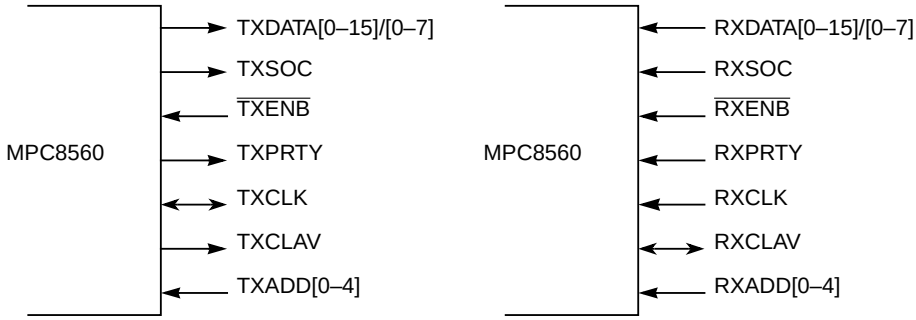


Figure 35-58. UTOPIA Slave Mode Signals

Table 35-45 describes UTOPIA slave mode signals.

Table 35-45. UTOPIA Slave Mode Signals

Signal	Description
TxDATA[0–15] /[0–7]	Transmit data bus. Carries transmit data from the ATM controller to the master device. TxDATA[15]/[7] is the msb, TxDATA[0] is the lsb.
TxSOC	Transmit start of cell. Asserted by an ATM controller as the first byte of a cell is sent on the TxDATA lines.
TxENB	Transmit enable. An input to the ATM controller. It is asserted by the UTOPIA master to signal the slave to send data in the next TxCLK cycle.
TxCLAV	Transmit cell available. Asserted by the ATM controller to indicate it has a complete cell to transmit.
TxPRTY	Transmit parity. Asserted by the ATM controller. It is an odd parity bit over the TxDATA.
TxCLK	Transmit clock. Provides the synchronization reference for the TxDATA, TxSOC, $\overline{\text{TxENB}}$, TxCLAV, and TxPRTY signals. All of the above signals are sampled at low-to-high transitions of TxCLK.
TxADD[0–4]	Transmit address. Address bus from the master to the ATM controller used to select the appropriate M-PHY device.
RxDATA[0–15]/ [0–7]	Receive data bus. Carries receive data from the master to the ATM controller. RxDATA[15]/[7] is the msb, RxDATA[0] is the lsb.
RxSOC	Receive start of cell. Asserted by the master device whenever the first byte of a cell is being received on the RxDATA lines.
RxENB	Receive enable. Asserted by the master device to signal the slave to sample the RxDATA and RxSOC signals.
RxCLAV	Receive cell available. Asserted by the ATM controller to indicate it can receive a complete cell.
RxPRTY	Receive parity. Asserted by the PHY device. It is an odd parity bit over the RxDATA[0–15]. If there is a RxPRTY error and the receive parity check FPSMR[RxP] is enabled, the cell is discarded. See see Section 35.13.2, “FCC Protocol-Specific Mode Register (FPSMR).”
RxCLK	Receive clock. Provides the synchronization reference for the RxDATA, RxSOC, $\overline{\text{RxENB}}$, RxCLAV, and RxPRTY signals. All the above signals are sampled at low-to-high transitions of RxCLK.
RxADD[0–4]	Receive address. Address bus from master to the ATM controller device used to select the appropriate M-PHY device.

35.12.2.1 UTOPIA Slave Multiple PHY Operation

In multiple PHY UTOPIA slave mode, cells are transferred using cell-level handshake as defined by the UTOPIA level-2 standard. The user should write the ATM controller PHY address in FPSMR[PHY ID].

35.12.2.2 UTOPIA Clocking Modes

The UTOPIA clock can be generated internally or externally. If the UTOPIA clock is to be generated internally, the user should assign one of the baud-rate generators to supply the UTOPIA clock. See Chapter 23, “CPM Multiplexing.”

35.12.2.3 UTOPIA Loop-Back Modes

The UTOPIA interface supports loop-back mode. In this mode, the Rx and Tx UTOPIA signals are shorted internally. Output pins are driven; input pins are ignored.

Note that in loop-back mode, the transmitter and receiver must operate in complementary modes. For example, if the transmitter is master, the receiver must be a slave (FPSMR[TUMS] = 0, FPSMR[RUMS] = 1).

Modes are selected through GFMR[DIAG], as shown in Table 35-46.

Table 35-46. UTOPIA Loop-Back Modes

DIAG	Description
00	Normal mode
01	Loop-back. UTOPIA Rx and Tx signals are shorted internally. Output pins are driven, input pins are ignored.
1x	Reserved

35.13 ATM Registers

The following sections describe the configuration of the registers in ATM mode.

35.13.1 General FCC Mode Register (GFMR)

The GFMR mode field should be programmed for ATM mode. To enable transmit and receive functions, ENT and ENR must be set as the last step in the initialization process. Full GFMR details are given in Section 34.2, “General FCC Mode Registers (GFMRx).”

35.13.2 FCC Protocol-Specific Mode Register (FPSMR)

The FCC protocol-specific mode register (FPSMR), shown in Figure 35-59, controls various protocol-specific FCC functions. The user should initialize the FPSMR. Erratic

behavior may result if there is an attempt to write to the FPSMR while the transmitter and receiver are enabled.

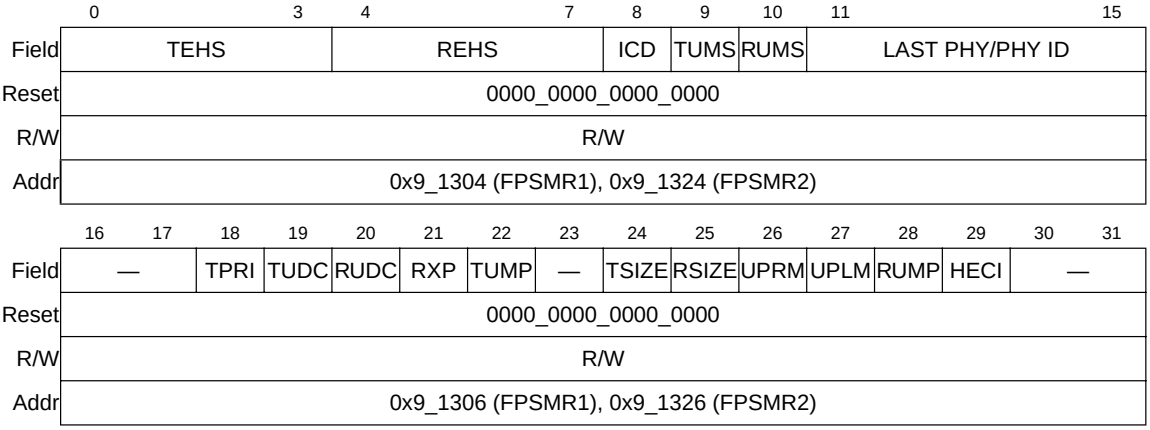


Figure 35-59. FCC ATM Mode Register (FPSMR)

Table 35-47 describes FPSMR fields.

Table 35-47. FCC ATM Mode Register (FPSMR)

Bits	Name	Description
0–3	TEHS	Transmit extra header size. Used only in user-defined cell mode to hold the Tx user-defined cells' extra header size. Values between 0–11 are valid. TEHS = 0 generates 1 byte of extra header; TEHS = 11 generates 12 bytes of extra header. Note: When working with a 16-bit UTOPIA interface, TEHS must represent an even number of header bytes (so the actual programmed value should be odd).
4–7	REHS	Receive extra header size. Used only in user-defined cell mode to hold the Rx user-defined cells' extra header size. Values between 0–11 are valid. For REHS = 0, the receiver expects 1 byte of extra header; for REHS = 11, it expects 12 bytes of extra header. Note: When working with a 16-bit UTOPIA interface, REHS must represent an even number of header bytes (so the actual programmed value should be odd).
8	ICD	Idle cells discard 0 Discard idle cells (GFC, VPI, VCI, PTI =0). 1 Do not discard idle cells.
9	TUMS	Transmit UTOPIA master/slave mode 0 Transmit UTOPIA master mode is selected. 1 Transmit UTOPIA slave mode is selected.
10	RUMS	Receive UTOPIA master/slave mode 0 Receive UTOPIA master mode is selected. 1 Receive UTOPIA slave mode is selected.
11–15	LAST PHY/ PHY ID	Last PHY. (Multiple PHY master mode only.) The UTOPIA interface polls all PHYs starting from PHY address 0 and ending with the PHY address specified in LAST PHY. (The number of active PHYs are LAST PHY+1). LAST PHY should be specified in both multiplex and direct-polling modes. PHY ID. (Multiple PHY slave mode only.) Determines the PHY address of the ATM controller when configured as a slave in a multiple PHY ATM port.
16–17	—	Reserved, should be cleared.

Table 35-47. FCC ATM Mode Register (FPSMR) (continued)

Bits	Name	Description
18	TPRI	Transmitter priority. Used to adjust the default priority of the FCC transmitter. It is strongly recommended to set TPRI when in multi-PHY mode; for other modes, it should remain cleared. 0 High priority (default operation) 1 Reduced priority (multi-PHY mode)
19	TUDC	Transmit user-defined cells 0 Regular 53-byte cells. 1 User-defined cells.
20	RUDC	Receive user-defined cells 0 Regular 53-byte cells. 1 User-defined cells.
21	RxP	Receive parity check. 0 Check Rx parity line. 1 Do not check Rx parity line.
22	TUMP	Transmit UTOPIA multiple PHY mode 0 Transmit UTOPIA single PHY mode is selected. 1 Transmit UTOPIA multiple PHY mode is selected.
23	—	Reserved, should be cleared.
24	TSIZE	Transmit UTOPIA data bus size 0 UTOPIA 8-bit data bus size. 1 UTOPIA 16-bit data bus size.
25	RSIZE	Receive UTOPIA data bus size 0 UTOPIA 8-bit data bus size. 1 UTOPIA 16-bit data bus size.
26	UPRM	UTOPIA priority mode. 0 Round robin. Polling is done from PHY zero to the PHY specified in LAST PHY. When a PHY is selected, the UTOPIA interface continues to poll the next PHY in order. 1 Fixed priority. Polling is done from PHY zero to the PHY specified in LAST PHY. When a PHY is selected, the UTOPIA interface continues to poll from PHY zero.
27	UPLM	UTOPIA polling mode. 0 Multiplex polling. Polling is done using RxAdd[0–4] and Clav[0]. Selection is done using RxAdd[0–4]. Up to 31 PHYs can be polled. 1 Direct polling. Polling is done using Clav[0–3]. Selection is done using RxAdd[0–2]. Up to 4 PHYs can be polled.
28	RUMP	Receive UTOPIA multiple PHY mode. 0 Receive UTOPIA single PHY mode is selected. 1 Receive UTOPIA multiple PHY mode is selected.
29	HECI	HEC included. Used in UDC mode only. 0 HEC octet is not included when UDC mode is enabled. 1 HEC octet is included when UDC mode is enabled.
30–31	—	Reserved, should be cleared.

35.13.3 ATM Event Register (FCCE)/ Mask Register (FCCM)

The FCCE register is the ATM controller event register when the FCC operates in ATM mode. When it recognizes an event, the ATM controller sets the corresponding FCCE bit.

Interrupts generated by this register can be masked in FCCM. FCCE is memory-mapped and can be read at any time. Bits are cleared by writing ones to them; writing zeros has no effect. Unmasked bits must be cleared before the CP clears the internal interrupt request.

FCCM is the ATM controller mask register. The FCCM has the same bit format as FCCE. Setting an FCCM bit enables and clearing a bit masks the corresponding interrupt in the FCCE.

Figure 35-60 depicts the ATM event register/FCC mask register.

	0–4	5	6	7	8	9	10	11	12	13	14	15
Field	—	TIRU	GRLI	GBPB	GINT3	GINT2	GINT1	GINT0	INTO3	INTO2	INTO1	INTO0
Reset	0000_0000_0000_0000											
R/W	R/W											
Addr	0x9_1310 (FCCE1), 0x9_1330 (FCCE2)/ 0x9_1314 (FCCM1), 0x9_1334 (FCCM2)											

	16–31
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x9_1312 (FCCE1), 0x9_1332 (FCCE2)/ 0x9_1316 (FCCM1), 0x9_1336 (FCCM2)

Figure 35-60. ATM Event Register (FCCE)/FCC Mask Register (FCCM)

Table 35-48 describes FCCE fields.

Table 35-48. FCCE/FCCM Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	TIRU	Transmit internal rate underrun. A transmit internal rate counter expired and a cell was not sent because the transmit FIFO was empty. TIRU may be set only when using transmit internal rate mode; see Section 35.13.8, “FCC Transmit Internal Rate Register (FTIRRx).”
6	GRLI	Global red-line interrupt. GRLI is set when a free buffer pool’s RLI flag is set. The RLI flag is also set in the free buffer pool’s parameter table.
7	GBPB	Global buffer pool busy interrupt. GBPB is set when a free buffer pool’s BUSY flag is set. The BUSY flag is also set in the free buffer pool’s parameter table.
8–11	GINT*x*	Global interrupt. Set when the number of events sent to the corresponding interrupt queue reaches the corresponding event threshold. See Section 35.11, “ATM Exceptions.”
12–15	INTO*x*	Interrupt queue overflow. Set when an overflow condition occurs in the corresponding interrupt queue. This occurs when the CP attempts to overwrite a valid interrupt entry. See Section 35.11.1, “Interrupt Queues.”
16–31	—	Reserved, should be cleared.

35.13.4FCC Transmit Internal Rate Mode

In internal rate mode the total transmission rate is the sum of the rates assigned for all PHYs. This register controls how internal rate is configured. In internal rate mode (GFEMR[TIREM] = 0), the internal rate assigned per PHY is configured by registers FTIRR[0-3]. In internal rate expanded mode (GFEMR[TIREM] = 1), registers FTIRR[0-3] control the available rates, but the PHY settings are configured in registers FIRPER, FIRSR_HI and FIRSR_LO. In TIREM = 0 mode internal rate can only be used for PHYs[0-3], whereas in TIREM = 1 mode up to 31 PHYs are supported. If TIREM = 1 mode is selected, the transmit internal rate underrun (TIRU) status per PHY may be read at any time in register FIRER.

35.13.5FCC Transmit Internal Rate Port Enable Register (FIRPER)

This register enables internal rate transmission for PHYs[0-30]. It is valid only if GFEMR[TIREM] = 1. If a PHY is not enabled in FIRPER, all TxClav indications from that PHY will be masked. The user should configure FIRPER according to the PHY addresses which are being used on the UTOPIA bus and should not enable PHYs with addresses larger then the last PHY address set by FPSMR[Last PHY]. PHYs can be enabled or disabled at any time—for example, if a TIRU event has occurred.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	PE0	PE1	PE2	PE3	PE4	PE5	PE6	PE7	PE8	PE9	PE10	PE11	PE12	PE13	PE14	PE15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_1380 (FIRPER1), 0x9_13A0 (FIRPER2)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	PE16	PE17	PE18	PE19	PE20	PE21	PE22	PE23	PE24	PE25	PE26	PE27	PE28	PE29	PE30	—
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_1382 (FIRPER1), 0x9_13A2 (FIRPER2)															

Figure 35-61. FCC Transmit Internal Rate Port Enable Register (FIRPER)

Table 35-49 describes FIRPER*x* fields.

Table 35-49. FIRPER*x* Field Descriptions (TIREM=1)

Bit	Name	Description
0–15	PEy	Port enable 0 Transmit internal rate for PHY address y is disabled. TxClav from this PHY is masked. 1 Transmit Internal rate for PHY address y is enabled. The rate assigned for PHY y is selected by register FIRSR_HI (refer to Section 35.13.7, “FCC Internal Rate Selection Registers (FIRSR_HI, FIRSR_LO)”).
16–30	PEy	Port enable. 0 Transmit internal rate for PHY address y is disabled. TxClav from this PHY is masked. 1 Transmit Internal rate for PHY address y is enabled. The rate assigned for PHY y is selected by register FIRSR_LO (refer to Section 35.13.7, “FCC Internal Rate Selection Registers (FIRSR_HI, FIRSR_LO)”).
31	—	Reserved, should be cleared.

35.13.6FCC Internal Rate Event Register (FIRER)

Transmit internal rate underrun (TIRU) errors are reported for any PHY that has a transmission deficiency of 8 cells. Under this condition and in internal rate mode only, FCCE[TIRU] is set, and if the corresponding bit in the FCC mask register (FCCM[TIRU]) is set, an interrupt is generated. If TIREM = 1, the TIRU status per PHY can be read at any time in the FCC internal rate event register (FIRER). Once FIRER[TIRUy] error status is set, it can be cleared only by writing 1 to it. To prevent an underrun PHY from continuously reporting errors, it can be disabled by FIRPER. The sequence of disabling a PHY is as follows:

- Disable PHY y by clearing FIRPER[y]
- Clear event FIRER[y] by writing 1 to it
- Clear event FCCE[TIRU] by writing 1 to it

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	TIRU 0	TIRU 1	TIRU 2	TIRU 3	TIRU 4	TIRU 5	TIRU 6	TIRU 7	TIRU 8	TIRU 9	TIRU 10	TIRU 11	TIRU 12	TIRU 13	TIRU 14	TIRU 15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_1384 (FIRER1), 0x9_13A4 (FIRER2)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	TIRU 16	TIRU 17	TIRU 18	TIRU 19	TIRU 20	TIRU 21	TIRU 22	TIRU 23	TIRU 24	TIRU 25	TIRU 26	TIRU 27	TIRU 28	TIRU 29	TIRU 30	—
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_1386 (FIRER1), 0x9_13A6 (FIRER2))															

Figure 35-62. FCC Internal Rate Event Register (FIRER)

Table 35-50 describes FIRER*x* fields.

Table 35-50. FIRER*x* Field Descriptions (TIREM=1)

Bit	Name	Description
0–30	TIRUy	Transmit internal rate underrun 0 There is no transmission underrun for this PHY. 1 Transmit internal rate underrun or PHY address y has occurred. Bit is cleared by writing 1 to it. Writing 0 has no effect on value.
31	—	Reserved, should be cleared.

35.13.7FCC Internal Rate Selection Registers (FIRSR_HI, FIRSR_LO)

If TIREM = 1, each PHY can be assigned one of four rates, as configured by the four FCC transmit internal rate timers. The FCC internal rate selection registers (FIRSRx_HI, FIRSRx_LO), shown in Figure 35-63 and Figure 35-64, assign rate group to each of the PHYs.

	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
Field	GS0	GS1	GS2	GS3	GS4	GS5	GS6	GS7
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x9_1388 (FIRSR1_HI), 0x9_13A8 (FIRSR2_HI))							

	16 17	18 19	20 21	22 23	24 25	26 27	28 29	30 31
Field	GS8	GS9	GS10	GS11	GS12	GS13	GS14	GS15
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x9_138A (FIRSR1_HI), 0x9_13AA (FIRSR2_HI)							

Figure 35-63. FCC Internal Rate Selection Register HI (FIRSR*x*_HI)

Table 35-51 describes FIRSR*x*_HI fields.

Table 35-51. IRSR*x*_HI Field Descriptions (TIREM=1)

Bit	Name	Description
0–31	GSy	Group select for PHY y 00The transmit internal rate for PHY address y is controlled by FTIRRx_GRP0. 01The transmit internal rate for PHY address y is controlled by FTIRRx_GRP1. 10The transmit internal rate for PHY address y is controlled by FTIRRx_GRP2. 11The transmit internal rate for PHY address y is controlled by FTIRRx_GRP3.

	0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15
Field	GS16	GS17	GS18	GS19	GS20	GS21	GS22	GS23
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x9_138C (FIRSR1_HI), 0x9_13AC (FIRSR2_HI)							

	16 17	18 19	20 21	22 23	24 25	26 27	28 29	30 31
Field	GS24	GS25	GS26	GS27	GS28	GS29	GS30	—
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x9_138E (FIRSR1_HI), 0x9_13AE (FIRSR2_HI)							

Figure 35-64. FCC Internal Rate Selection Register LO (FIRSR*x*_LO)

Table 35-52 describes FIRSR*x*_LO fields.

Table 35-52. FIRSR*x*_LO Field Descriptions (TIREM=1)

Bit	Name	Description
0-29	GSy	Group select for PHY y 00The transmit internal rate for PHY address y is controlled by FTIRRx_GRP0. 01The transmit internal rate for PHY address y is controlled by FTIRRx_GRP1. 10The transmit internal rate for PHY address y is controlled by FTIRRx_GRP2. 11The transmit internal rate for PHY address y is controlled by FTIRRx_GRP3.
30–31	—	Reserved, should be cleared.

35.13.8FCC Transmit Internal Rate Register (FTIRR*x*)

If GFEMR[TIREM] = 0, PHYs at addresses 0–3 have their own FCC transmit internal rate registers (FTIRRx_PHY0–FTIRRx_PHY3) for use in transmit internal rate mode. If TIREM=1, FTIRRx are used as group timers and PHYs at addresses 0-30 are assigned to a rate group by FIRSRx_HI and FIRSRx_LO. FTIRR*x*, shown in Figure 35-67, includes the initial value of the internal rate timer. The clock to the internal rate timers is supplied by one of four baud-rate generators selected in CMXUAR. Note that in slave mode, FTIRR0 is used regardless of the slave PHY address.

	0	1 – 7
Field	TRM	Initial Value
Reset	0000_0000	
R/W	R/W	
Address	GFEMR[TIREM=0] FCC1: 0x9_131C (FTIRR1_PHY0), FCC1: 0x9_131D (FTIRR1_PHY1), FCC1: 0x9_131E (FTIRR1_PHY2), FCC1: 0x9_131F (FTIRR1_PHY3), FCC2: 0x9_133C (FTIRR2_PHY0), FCC2: 0x9_133D (FTIRR2_PHY1), FCC2: 0x9_133E (FTIRR2_PHY2), FCC2: 0x9_133F (FTIRR2_PHY3).	GFEMR[TIREM=1] FCC1: 0x9_131C (FTIRR1_GRP0), FCC1: 0x9_131D (FTIRR1_GRP1), FCC1: 0x9_131E (FTIRR1_GRP2), FCC1: 0x9_131F (FTIRR1_GRP3), FCC2: 0x9_133C (FTIRR2_GRP0), FCC2: 0x9_133D (FTIRR2_GRP1), FCC2: 0x9_133E (FTIRR2_GRP2), FCC2: 0x9_133F (FTIRR2_GRP3).

Figure 35-65. FCC Transmit Internal Rate Register (FTIRR)

Table 35-55 describes FTIRR*x* fields.

Table 35-53. FTIRR*x* Field Descriptions

Bit	Name	Description
0	TRM (TIREM=0)	PHY transmit mode 0 External rate mode. 1 Internal rate mode.
	TRM (TIREM=1)	Group transmit mode 0 Group rate timer [x] disabled. 1 Internal rate timer for Group[x] is enabled and division factor is set by Initial Value field.
1–7	Initial Value	The initial value of the internal rate timer. A value of 0x7F produces the minimum clock rate (BRG CLK divided by 128); 0x00 produces the maximum clock rate (BRG CLK divided by 1).

Figure 35-68 shows how transmit clocks are determined.

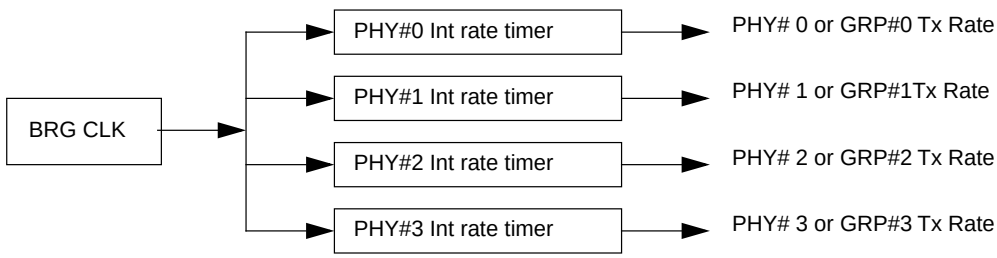


Figure 35-66. FCC Transmit Internal Rate Clocking

35.13.8.1 Example

If the MPC8280 is connected to four 155 Mbps PHY devices and the maximum transmission rate is 155 Mbps for the first PHY and 10 Mbps for the rest of the PHYs, the BRG CLK should be set according to the highest rate. If the system clock is 133 MHz, the

BRG should be programmed to divide the system clock by 362 to generate cell transmit requests every 362 system clocks:

$$\frac{(133\text{MHz} \times (53 \times 8))}{155.52\text{Mbps}} = 362$$

For the 155 Mbps PHY, the FTIRR divider should be programmed to zero (the BRG CLK is divided by one); for the rest of the 10 Mbps PHYs, the FTIRR divider should be programmed to 14 (the BRG CLK is divided by 15).

35.13.9 Internal Rate Programming Model

The programming sequence in TIREM = 0 mode is as follows:

1. Clear GFEMRx[TIREM]
2. Program FTIRRx

The programming sequence in TIREM = 1 mode is as follows:

1. Clear FTIRRx[TRM]
2. Set GFEMRx[TIREM]
3. Program FIRSRx_HI and FIRSRx_LO
4. Program FTIRRx
5. Program FIRPERx

If FTIRRx are set to generate same order of magnitude rates, setting round robin polling mode is more adequate than fixed priority mode. To reduce the risk of transmit underrun if there are a few PHYs with high internal rate and a number of PHYs with a low internal rate, the fast PHYs should be assigned consecutive addresses starting at 0 and fixed priority mode should be chosen.

35.14 ATM Transmit Command

The CPM command set includes an ATM TRANSMIT that can be sent to the CP command register (CPCR), described in Section 20.4.1.

The ATM TRANSMIT command (CPCR[opcode] = 0b1010, CPCR[SBC[code]] = 0b01110, CPCR[SBC[page]] = 0b00100 or 0b00101 (FCC1 or FCC2), CPCR[MCN] = 0b0000_1010) turns a passive channel into an active channel by inserting it into the APC scheduling table. Note that an ATM TRANSMIT command should be issued only after the channel's TCT is completely initialized and the channel has BDs ready to transmit. Note also that CPCR[SBC[code]] = 0b01110 and not FCC1 or FCC2 code.

Before issuing the command, the user should initialize COMM_INFO fields in the parameter RAM as described in Figure 35-67.

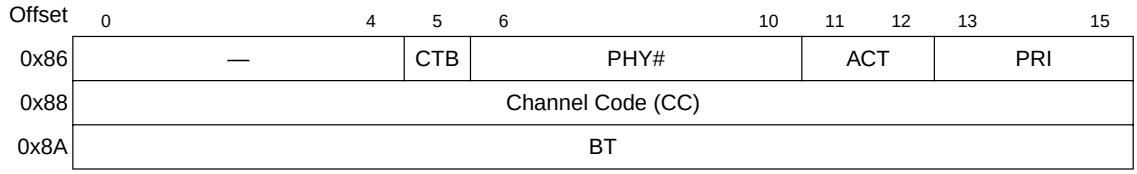


Figure 35-67. COMM_INFO Field

Table 35-54 describes COMM_INFO fields.

Table 35-54. COMM_INFO Field Descriptions

Offset	Bits	Name	Description
0x86	0–4	—	Reserved, should be cleared.
	5	CTB	Connection tables bus. Used for external channels only 0 External connection tables reside on the system bus. 1 External connection tables reside on the local bus.
	6–10	PHY#	PHY number. In single PHY mode this field should be cleared In multiple PHY mode this field is an index to the APC parameter table associated with this channel.
	11–12	ACT	ATM channel type 00 Other channel 01 VBR channel 1x Reserved
	13-15	PRI	APC priority level. 000—highest priority (APC_LEVEL1), 111—lowest priority (APC_LEVEL8).
0x88	0-15	CC	Channel code. The channel code associated with the current channel.
0x8A	0-15	BT	Burst tolerance. For use by VBR channels only (ACT field is 0b01). Specifies the initial burst tolerance (GCRA burst credit) of the current VC.

35.15 SRTS Generation and Clock Recovery Using External Logic

The MPC8560 supports SRTS generation using external logic. If SRTS generation is enabled (TCT[SRT] = 1), the MPC8560 reads SRTS[0–3] from the external SRTS logic and inserts it into 4 cells whose SN fields equal 1, 3, 5, and 7, as shown in Figure 35-68.

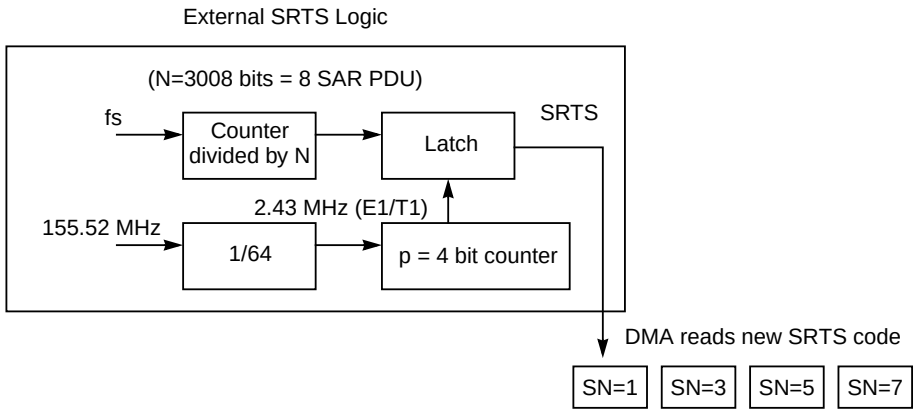


Figure 35-68. AAL1 SRTS Generation Using External Logic

For every eight cells, the external SRTS logic should supply a valid SRTS code. The CP reads the SRTS code from the bus selected in TCT[BIB] using a DMA read cycle of 1-byte data size. Each AAL1 channel can be programmed to select one of 16 addresses available for reading the SRTS result. The SRTS code should be placed on the least-significant nibble of that address (SRTS[0]=lsb, SRTS[3]=msb). The SRTS is synchronized with the sequence count cycle—SRTS[0] is inserted into the cell with SN = 7; SRTS[3] is inserted into the cell with SN = 1. For every eighth AAL1 SAR PDU, the SRTS logic samples a new SRTS and stores it internally. The SRTS is a sample of a 4-bit counter with a 2.43-MHz reference clock (for E1/T1) synchronized with the network clock.

The MPC8560 supports clock recovery using an external SRTS PLL. If SRTS recovery is enabled (RCT[SRT]=1), the MPC8560 tracks the SRTS from four incoming cells whose SN field equals 1, 3, 5, and 7 and writes the result to external SRTS logic, as shown in Figure 35-69.

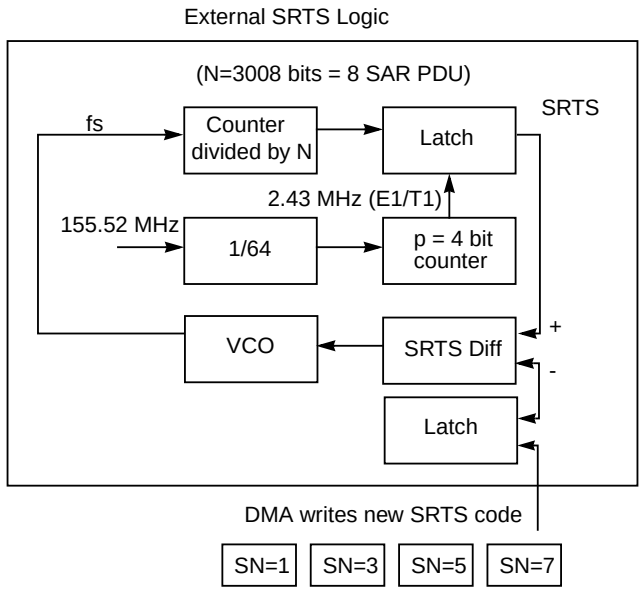


Figure 35-69. AAL1 SRTS Clock Recovery Using External Logic

On every eighth cell, the MPC8560 writes a new SRTS code to the external logic using the bus selected in RCT[BIB]. The CP writes the SRTS code using a DMA write cycle of 1-byte data size. Each AAL1 channel can be programmed to select one of 16 addresses available for writing the SRTS result. The SRTS code is written to the least-significant nibble of that address (SRTS[0]=lsb, SRTS[3]=msb). The SRTS is synchronized with the sequence count cycle—SRTS[3] is read from the cell with SN = 1 and SRTS[0] is read from the cell with SN = 7. The SRTS PLL makes periodic clock adjustments based on the difference between a locally generated SRTS and a remotely generated SRTS retrieved every eight received cells.

35.16 Configuring the ATM Controller for Maximum CPM Performance

The following sections recommend ATM controller configurations to maximize CPM performance.

35.16.1 Using Transmit Internal Rate Mode

When the total transmit rate is less than the PHY rate, use the transmit internal rate mode and configure the internal rate clock to the maximum bit rate required. (See 35.2.1.4, “Transmit External Rate and Internal Rate Modes.”) The PHY then automatically fills the unused bandwidth with idle cells, not the ATM controller. If the internal rate mode is not used, CPM performance is consumed generating the idle cell payload and using the scheduling algorithm to fill the unused bandwidth at the higher PHY rate.

For example, suppose a system uses a 155.52-Mbps OC-3 device as PHY0, but the maximum required data rate is only 100 Mbps. In transmit internal rate mode, the user can configure the internal rate mechanism to clock the ATM transmitter at a cell rate of 100 Mbps. If the system clock is 133 MHz, program a BRG to divide the system clock by 563 to generate a transmit cell request every 563 CPM clocks:

$$\frac{(133\text{MHz} \times (53 \times 8))}{100\text{Mbps}} = 563$$

Set FTIRR*x*_PHY0[TRM] to enable the transmit internal rate mode and clear FTIRR*x*_PHY0[Initial Value] since there is no need to further divide the BRG. See Section 35.13.8, “FCC Transmit Internal Rate Register (FTIRRx).”

In external rate mode, however, the transmit cell request frequency is determined by the PHY’s maximum rate, not by internal FCC counters. If an OC-3 PHY is used with the ATM controller in external rate mode, the requests must be generated every 362 CPM clocks (assuming a 133-MHz CPM clock). If only 100 Mbps is used for real data, 36% of the transmit cell requests consume CPM processing time sending idle cells.

35.16.2 APC Configuration

Maximizing the number of cells per slot (CPS) and minimizing the priority levels defined in the APC data structure improves CPM performance:

- Cells per slot. CPS defines the maximum number of ATM cells allowed to be sent during a time slot. (See Section 35.3.3.1, “Determining the Cells Per Slot (CPS) in a Scheduling Table.”) The scheduling algorithm is more efficient sending multiple cells per time slot using the linked-channel field. Therefore, choose the maximum number of cells per slot allowed by the application.
- Priority levels. The user can configure the APC data structure to have from one to eight priority levels. (See Section 35.3.6, “Determining the Priority of an ATM Channel.”) For each time slot, the scheduling algorithm scans all priority levels and maintains pointers for each level. Therefore, enable only the minimum number of priority levels required.

35.16.3 Buffer Configuration

Using statically allocated buffers of optimal sizes also improves CPM performance:

- Buffer size. Opening and closing buffer descriptors consumes CPM processing time. Because smaller buffers require more opening and closing of BDs, the optimal buffer size for maximum CPM performance is equal to the packet size (an AAL5 frame, for example).
- Free buffer pool. When the free buffer pool is used, the CPM dynamically allocates buffers and links them to a channel’s BD. In static buffer allocation, the core assigns

a fixed data buffer to each BD. (See Section 35.10.5.2, “Receive Buffer Operation.”) When allowed by the application, use static buffer allocation to increase CPM performance.

35.16.4 Compression and Connection Tables

For systems seeking high performance ATM capabilities, placing address compression tables (VP, if applicable, and VC) and connection tables (RCT/TCT) for external connections on the local bus will improve CPM performance:

- CAMs. CAMs can only be connected to the MPC8560 using the local bus.
- Local Bus Optimization. Because the local bus has been optimized to provide low-latency access to the CPM, placement of compression and connection tables are preferred on the local bus. Placement elsewhere (e.g. DDR SDRAM) would reduce performance. If the system is not designed for high performance ATM, placement of the tables in the DDR SDRAM puts substantial load on the DDR bus, and this loading must be accounted for by the system designer. It should be noted that performance calculations for the DDR bus must use 32-byte transactions regardless of the actual transaction size (VC table lookups of two bytes still require 32-byte performance calculations). The system designer must also be careful not to add up the bandwidth of transactions in performance calculations on the DDR bus (1000 VC lookups of two bytes each require 1000 DDR memory transactions, not 1000/16).

35.16.5 Level 2 Cache Utilization

For high performance ATM systems, it is strongly recommended that interrupt queues, free buffer pools, and buffer descriptors be stored as much as possible in the L2/private SRAM to increase CPM ATM performance:

- For smaller descriptors, cache lines can be locked here with that content, and the L2 can be used at its maximum size of 256-Kbytes.
- For larger descriptors, it may be appropriate to partition part of the L2 memory complex (128-Kbytes) as memory mapped SRAM and storing the structures there, effectively reducing the L2 cache size to 128-Kbytes.

Chapter 36
AAL1 Circuit Emulation Service

This chapter describes the implementation of circuit emulation service (CES) using AAL1 on the MPC8560.

36.1 Features

The AAL1 and CES features on the MPC8560 are as follows:

- AAL1
  - — Reassembly
    - – Reassembles PDU directly to external memory.
    - – Supports partially filled cells (configurable on a per-VC basis).
    - – Sequence number (SN) protection (CRC-3 and parity) check.
    - – Implements a 3-step SN algorithm.
    - – Detects and handles lost or misinserted cell.
    - – Maintains bit count integrity (dummy cell insertion).
    - – Dummy cell contents can be programmed by the user (per FCC).
    - – Pointer verification in structured AAL1 cell format.
    - – Automatic synchronization using the structured pointer during reassembly.
    - – SRTS (synchronous residual time stamp) gathering on every cycle (8 cells) in unstructured AAL1.
    - – Statistics gathering on a per-VC basis:
      - AAL1 valid cell count.
      - AAL1 lost cell count.
      - AAL1 misinserted cell count.
      - AAL1 pointer mismatch/parity error.
      - AAL1 Rx buffer pre-overflow.
  - — Segmentation
    - – Segment PDU directly from external memory.

  - Partially filled cells support (configurable on a per-VC basis).
  - Sequence number generation.
  - Sequence number protection (CRC-3 and even parity) generation.
  - Pointer generation during segmentation in structure AAL1 cell format.
  - Clock recovery using external SRTS logic during reassembly in unstructured AAL1.
  - Statistics gathering on a per-VC basis:
    - AAL1 Tx cell count.
    - AAL1 Tx buffer underrun.

- CES
  — ATM to TDM
    - Structured and unstructured data are transferred between the ATM and MCC automatically without CPU intervention.
    - In case of pre-underrun, the MCC start sending the last frame or the user-defined underrun template. The MCC and ATM controller automatically perform slip control with no CPU intervention.
    - In case of pre-overrun, the ATM receiver discards incoming cells until the MCC transmitter empties enough buffers for the receiver to restart.
    - Supports common channel signaling (CCS).
    - Supports channel associated signaling (CAS)
      - Up to four (one per trunk) CAS blocks residing in internal RAM when in automatic CAS mode and up to 8 blocks when in core CAS modify mode.
      - Supports N x 64 E1/T1 channels.
      - CAS routing table on per-VC basis.
      - Automatic unpacking of the CAS information during reassembly and updating of the internal CAS block.
  — TDM to ATM
    - Structured and unstructured data are automatically transferred between the MCC and ATM controller without CPU intervention.
    - Supports common channel signaling (CCS).
    - Supports channel associated signaling (CAS)
      - Up to four (one per trunk) CAS blocks residing in internal RAM when in automatic CAS mode and up to 8 blocks when in core CAS modify mode.
      - Support N x 64 E1/T1 channels.
      - CAS routing table on per-VC basis.

- Automatic packing of the CAS information from internal RAM to the AAL1 Tx cells.
- Once per superframe, the CPM fetches the CAS information from an external framer and updates each internal CAS block.

36.2 AAL1 Transmitter Overview

The MPC8560 supports both structured and unstructured AAL1 cell formats. For unstructured format, the transmitter reads 47 bytes from the external buffer and inserts them into the AAL1 user data field. For structured format, the transmitter reads 46 or 47 bytes from the external buffer and inserts them into the AAL1 user data field.

36.2.1 Data Path

The AAL1 PDU header, which consists of the sequence number (SN) and the sequence number protection (SNP) (CRC-3 and parity bit), is generated and inserted into the AAL1 Tx cell, as shown in Figure 36-1.

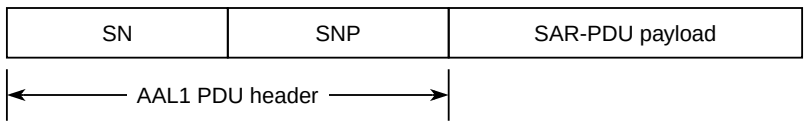


Figure 36-1. AAL1 Transmit Cell Format

When the transmitter operates in structured data transfer (SDT) mode, two types of AAL1 cells are defined: P (pointer) format and non-P format. The two formats are shown below.

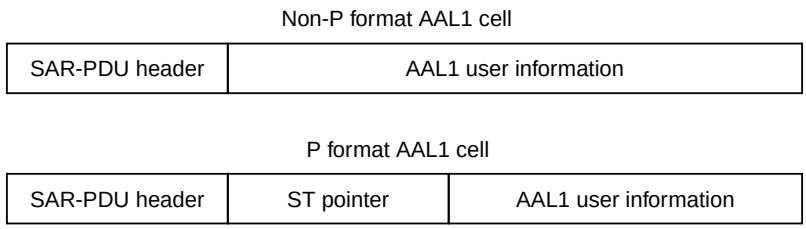


Figure 36-2. Non-P Format AAL1 Cell

The transmitter generates the structured pointer according to the I.363.1 ITU standard and inserts the pointer exactly once every cycle (eight successive cells). At the first opportunity, the transmitter inserts the structured pointer into a cell with an even sequence count (SC). When the end of the structure is not present in the current cycle, a P-format cell with a dummy pointer (127) is inserted into the last cell (SC = 6) of the cycle.

The MPC8560 supports partially filled cells configured on a per-VC basis. In this mode, only the valid octets are filled with user information; the rest of the cell is filled with padding octets.

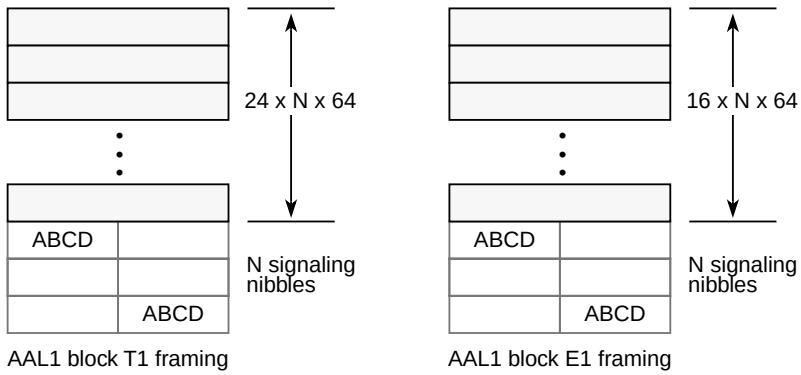
The MPC8560 supports synchronous residual time stamp (SRTS) generation using an external PLL. If this mode is enabled, the MPC8560 reads the SRTS code from external logic and inserts it into four outgoing cells. See Section 36.17, “SRTS Generation and Clock Recovery Using External SRTS Logic.”

36.2.2 Signaling Path

The MPC8560 automatically handles the signaling information as part of the interworking function. The ATM transmitter packs the signaling information at the end of each superframe during the data segmentation process. Each VC is associated to one signaling block by an internal routing table; see Section 36.4.6, “Channel Associated Signaling (CAS) Support.” The signaling information that resides in the internal RAM is inserted into the AAL1 cell according to the AF-VTOA-0078 specification.

The AAL1 structure is divided into two sections. The first section carries the N x 64 payload, and the second carries the signaling bits that are associated with the payload.

The MPC8560 supports two framing modes: one for N x 64 DS1 ESF (extended superframe) framing, and the other for N x 64 E1 G.704 framing. See Figure 36-3.. Each of the internal signaling blocks can be used to deliver only one of the framing formats; that is, they cannot be changed dynamically.



Note: The CAS block size is (N + 1) nibble if N is an odd number.

Figure 36-3. AAL1 Framing Formats

36.3 AAL1 Receiver Overview

The ATM controller supports both AAL1 structured and unstructured formats. For the unstructured format, 47 octets are copied to the current receive buffer and for the structured format, 46 (P format) or 47 (non-P format) octets are copied to the current receive buffer.

The AAL1 PDU header, which consists of the sequence number (SN) and the sequence number protection (SNP) (CRC-3 and parity bit), is checked and the result is delivered to the 3-step-SN algorithm. The 3-step-SN algorithm (see Figure 36-15., “Pre-Overrun

Sequence") handles the lost or misinserted cells. This algorithm can detect one lost or misinserted cell and maintain synchronization. If more than one cell is lost or misinserted, the 3-step-SN algorithm switches to hunt mode where it stays until a cell with a valid SN field is received. After the receiver switches to hunt mode, it closes the RxBD, modifies the receive statistics, generates an optional interrupt to the CPU and performs a restart sequence.

The restart sequence is implemented only when the ATM channel works in CES mode (RCT[CESM]). In CAS mode (RCT[CASM]), the ATM receiver channel begins the restart sequence by dropping all incoming cells and advancing to the beginning of the next super frame, which is the first BD after the one marked with EOSF (end of super frame). When this BD is ready and the adaptive counter reaches the ATM_Start threshold, the receiver's write pointer is not longer in danger of overrunning the read pointer of the MCC transmitter; that is, it is safe to begin receiving cells again. The ATM receiver then begins the resynchronization process: for unstructured AAL1 type the ATM receiver waits for the first valid cell, and for structured AAL1 type the receiver waits for the first valid cell that contains a valid pointer. The first received octet becomes the first byte of the new BD (new super frame). (See 36.5, "ATM-to-TDM Adaptive Slip Control" and 36.4, "Interworking Functions.")

Note that when the ATM channel is not in CES mode, no restart sequence is performed; the ATM receiver immediately starts hunting for the first valid cell. The first received octet becomes the first byte of the next BD.

During reassembly, the ATM receiver channel's 3-step-SN algorithm status is delivered to the pointer verification mechanism. (See 36.7, "Pointer Verification Mechanism.") If the receiver operates in unstructured data format, the 3-step-SN algorithm status is delivered directly to the bit count integrity module. When partially filled cells arrive, the bit count integrity module copies only the valid octets from the received cell (or from the dummy cell if that cell is lost) to the current receive buffer.

In unstructured AAL1 format, when the receive process begins, the receiver hunts for the first cell with a valid sequence number (SN field). When one arrives, the receiver leaves the hunt state and begins receiving incoming cells.

In structured AAL1 format, when the receive process begins, the receiver hunts for the first cell with a valid structured pointer (not a dummy pointer), that points to the start of a new structure. When one arrives, the receiver leaves the hunt state, opens a new buffer and begins receiving. The structured pointer points to the first octet of the structured block, which then becomes the first byte of the new buffer.

During the reassembly process, if the receiver switches to hunt mode (due to the 3-step-SN algorithm or due to receiving two successive mismatched pointers), the ATM receiver closes the current RxBD, discards incoming cells, modifies the receive statistics accordingly and initiates a restart sequence. The receiver then waits for a cell with a valid structured pointer to regain synchronization and start receiving incoming cells again.

The MPC8560 supports SRTS clock recovery using an external PLL. The MPC8560 tracks the SRTS from the four incoming cells and writes the SRTS code to external logic. See Section 36.17, “SRTS Generation and Clock Recovery Using External SRTS Logic.” The data flow for an AAL1 receiver is summarized in Figure 36-4.

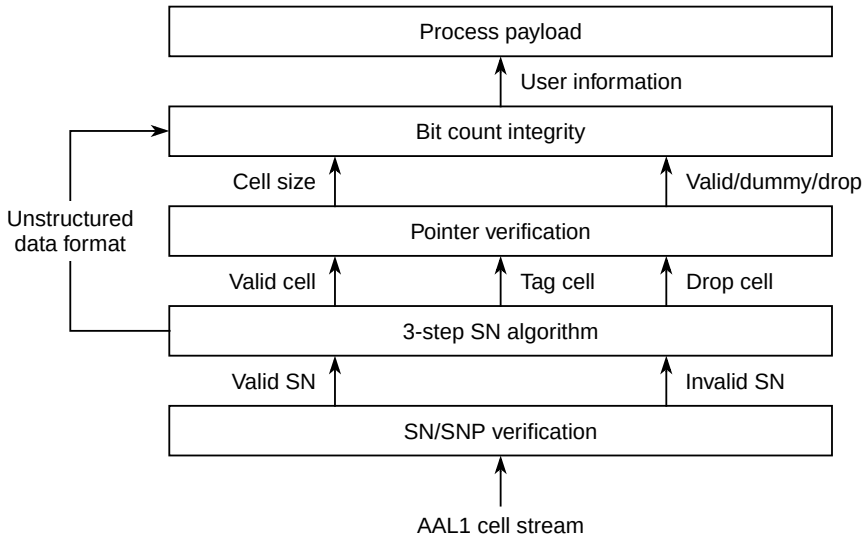


Figure 36-4. AAL1 Receiver Data flow

36.4 Interworking Functions

The MPC8560 supports the interworking of ATM and TDM. The TDM data processing is done by the MCC and the SI. (See the multi-channel controller (MCC) chapter and the serial interface (SI) chapter.) The ATM controller processes the ATM data.

Data forwarding between the ATM controller and the MCC can be done in two ways:

- Automatic data forwarding. This mode enables automatic data forwarding between AAL1 and transparent mode over a TDM.
- Core intervention. When an MCC receive buffer is full and its RxBD is closed, the MCC interrupts the core. The core copies the MCC’s receive buffer pointer to an ATM TxBD and sets the ready bit (TxBD[R]). Similarly, when an ATM receive buffer is full and its RxBD is closed, the core services the ATM controller’s interrupt by copying the ATM receive buffer pointer to an MCC TxBD and setting TxBD[R]. This mode is useful when additional core processing is required.

Possible interworking applications include the following:

- Circuit emulation service (CES)
- Carrying voice over ATM
- Multiplexing several low speed services, such as voice and data, onto one ATM connection

36.4.1 Automatic Data Forwarding

The basic concept of automatic data forwarding is to program the ATM controller and the MCC to process the same BD table (ring).

The MCC and ATM receivers must be programmed to operate in opposite E-bit polarity. That is, both receivers receive into buffers in which RxBD[E] is clear, and then set RxBD[E] when the buffer is full. For the ATM receiver, set RCT[INVE] of the AAL1-specific areas of the receive connection table; see Section 36.9.1, “Receive Connection Table (RCT).” For the MCC receiver, set CHAMR[EP]; see the multi-channel controller (MCC) chapter.

36.4.1.1 ATM-to-TDM

When going from ATM to TDM (shown in Figure 36-5Figure 36-5.), the ATM receiver reassembles data received from a particular channel to a specific BD table. The MCC transmitter is programmed to operate on the same table. When the ATM controller fills a receive buffer, the MCC controller sends it. The controllers synchronize on the ATM controller’s RxBD[E] and the MCC’s TxBD[R].

A threshold mechanism for the MCC transmitter is used to synchronize the automatic start of the ATM-to-TDM data forwarding. The transmitter waits until the start threshold is reached (enough cells are received) before sending the valid data. In effect, the MCC start threshold mechanism implements a jitter buffer designed to accommodate the CDV of the ATM network.

In a CES application, software should first initialize the MCC transmitter (see the MCC chapter), then initialize the ATM receiver. This way the MCC transmitter sends idle data (0xFF) until the ATM receiver fills enough buffers to reach the MCC start threshold. (The receiver is effectively filling a jitter buffer.) When the CES_Adaptive_Counter (CESAC) reaches the MCC_Start threshold, the MCC super channel polling CESAC starts sending the received data with the first frame SYNC. (The first octet using the first BD is sent on the first assigned time slot.) Figure 36.5 shows the flow of ATM-to-TDM interworking.

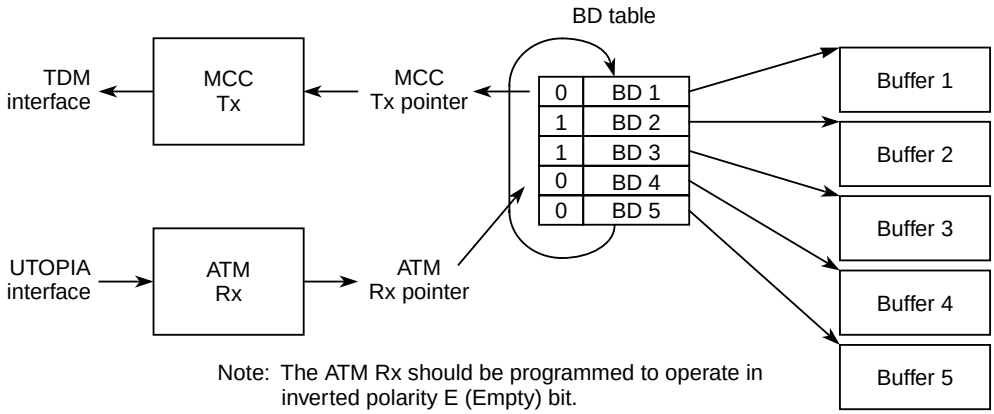


Figure 36-5. ATM-to-TDM Interworking

36.4.1.2 TDM-to-ATM

When going from TDM to ATM (depicted in), the MCC receiver routes data from the TDM line to a specific BD table. The ATM transmitter is programmed to operate on the same BD table. When the MCC fills a receive buffer, the ATM controller sends it. The two controllers synchronize on the MCC's RxBD[E] and the ATM controller's TxBD[R].

In a CES application, first initialize the ATM transmitter, then initialize the MCC receiver (see the MCC chapter) and set CHAMR[EP].

In order to prevent an overrun condition on the MCC receiver, the ATM transmitter should be programmed to work at a faster rate than the MCC super channel. This ensures that the ATM channel polls the common BD table at a higher rate than it is being filled by the MCC. In CES mode, the ATM controller ignores BSY (busy) events when the ATM tries to open a buffer that is not yet full; it continues polling the BD until the buffer is filled by the MCC receiver and then updates the relevant statistics; see 36.15, "Internal AAL1 Statistics Tables."

Note that if an overrun condition occurs on the MCC, despite the above mechanism, software should restart the MCC and ATM channels in order to recover.

Figure 36-6 shows the flow of TDM-to-ATM interworking.

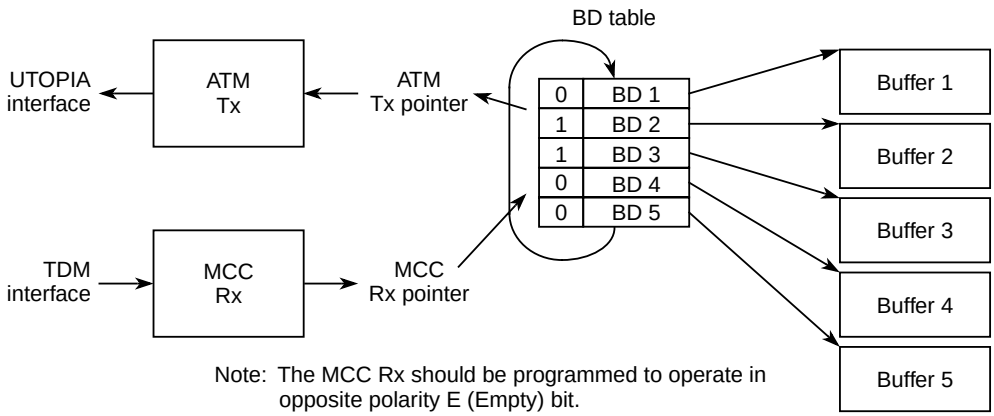


Figure 36-6. TDM-to-ATM Interworking

36.4.2 Timing Issues

Use of the TDM interface assumes that all communicating entities are synchronized; that is, that they are using a synchronized serial clock. If the TDM interfaces are not synchronized, a slip can occur in the reassembly buffer. In order to prevent the overrun and underrun condition, the MPC8560 maintains an adaptive slip control using a set of four threshold pointers and a counter for each ATM-TDM (VC to super channel) connection.

Before a buffer-not-ready event (ATM-to-TDM data forwarding) occurs at the MCC transmitter, the MCC buffer pointer reaches the MCC_Stop threshold. Consequently, the MCC pointer freezes on the last transmitted BD and starts sending the underrun template (or the last transmitted frame). In the meantime, the ATM receiver continues to write valid data and advance the ATM buffer pointer. When the adaptive counter CESAC reaches the MCC_start threshold and the MCC has finished sending a multiple frame size, the MCC exits the pre-underrun state, starts sending the valid received data and advances the MCC buffer pointer. (See 36.5, “ATM-to-TDM Adaptive Slip Control.”)

The same mechanism is implemented on the ATM side. When the ATM receiver (ATM-to-TDM data forwarding) reaches the ATM_Stop threshold (pre-overrun), the ATM controller switches to hunt mode and discards the channel’s incoming cells. In the meantime, the MCC transmitter continues to send valid data and advance the MCC buffer pointer. When CESAC falls to the ATM_start threshold, the ATM write pointer is advanced to the first BD after the one marked with EOSF (in CAS mode). When this BD is ready and CESAC reaches the ATM_Start threshold, the receiver’s write pointer is not longer in danger of overrunning the read pointer of the MCC transmitter; that is, it is safe to begin receiving cells again. The ATM receiver then begins the resynchronization process: for unstructured AAL1 type the ATM receiver waits for the first valid cell, and for structured AAL1 type the receiver waits for the first valid cell that contains a valid pointer. The first received octet becomes the first byte of the new BD (new super frame). (See 36.5, “ATM-to-TDM Adaptive Slip Control.”)

Note that when the ATM receiver is in hunt mode due to one of the following:

- Sequence number protection error (SNPE)
- Sequence count error (SCE)
- Structured pointer error (SPE)
- Slip condition

The signaling information (CAS) and SRTS information is not updated by the ATM controller until the ATM receiver switches to SYNC mode, that is, a valid cell is received in unstructured cell format or a valid pointer is received in structured cell formats.

Software should distinguish between the two types of overrun and underrun conditions:

6. The MCC and ATM controller can automatically recover from overruns and underruns caused by slips without any CPU intervention. (See Section 36.5, “ATM-to-TDM Adaptive Slip Control.”)
7. Global underrun (MCCE[GUN]) and overrun (MCCE[GOV]) conditions are errors that need CPU intervention because it is not known which channels are affected. The CPU should accordingly reinitialize the transmit parameters and/or the receive parameters to recover. (See the MCC chapter.)

36.4.3 Clock Synchronization (SRTS, Adaptive FIFO)

Clock synchronization methods, such as SRTS and adaptive FIFO, may be used to prevent reassembly buffer slip. The SRTS method may be implemented using external logic. The MPC8560 can read the SRTS from external logic and insert it into outgoing AAL1 cells and conversely, can track the SRTS from incoming AAL1 cells and deliver it to external logic. See Section 36.17, “SRTS Generation and Clock Recovery Using External SRTS Logic.”

Alternatively, an adaptive FIFO method can be implemented under core control. Adaptive FIFO is a way to hold the bridging buffer at its mid-level point. One way to implement it is to periodically poll the adaptive counter CESAC (difference between the MCC and ATM data pointers) and use this difference as a pseudo-SRTS; see 36.5.3, “CES Adaptive Threshold Tables.” Writing the pseudo-SRTS to the same external PLL logic used in the SRTS method adjusts the TDM clock.

36.4.4 Mapping TDM Time Slots to VCs

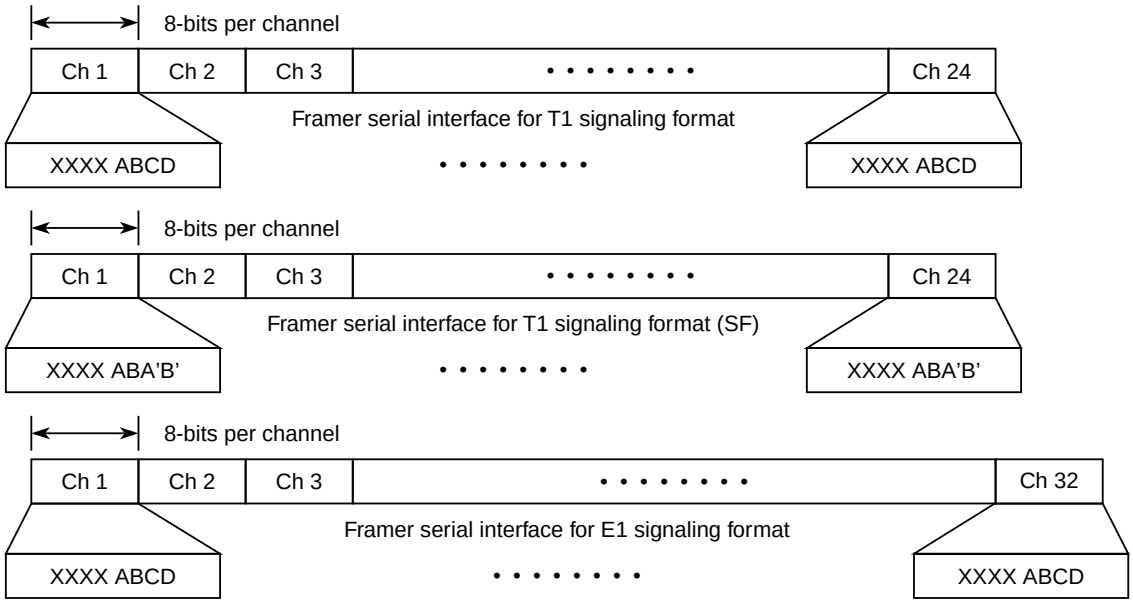
Any TDM time-slot combination can be routed to a specific data buffer using the MCC and its SI. (See the MCC chapter, and the serial interface (SI) chapter.) A common set of data buffers (one BD table) should be used by the ATM controller to route both the receive and transmit data. For information about ATM buffers see Section 36.11, “Buffer Descriptors.”

36.4.5 Trunk Condition

According to the Bellcore standard, the interworking function should be able to transmit special payloads on both the ATM and TDM channels to signal alarm conditions (bellcore TR-NWT-000170). The trunk condition can be generated under core control. The core may deliver buffers containing special data (trunk condition payload) to the ATM controller or MCC or even overwrite data buffers being used by the channels.

36.4.6 Channel Associated Signaling (CAS) Support

For applications requiring channel associated signaling (CAS) support, the CAS manipulation is done by an external framer. The MCC should be programmed to receive or transmit the internal CAS block transparently through the external framer's serial interface; see Figure 36-7. The internal CAS block (depicted in Figure 36-8.) can be adjusted to comply with a specific framer's serial interface. (See 36.4.7.1, "CAS Routing Table.")

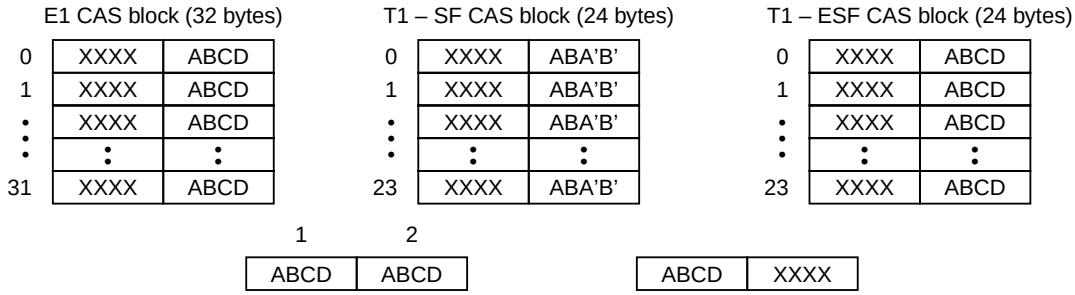


Note: The MCC should capture the signaling data on the last frame of a super frame. See Section 1.4.7.2, "TDM-to-ATM CAS Support".

Figure 36-7. Mapping CAS Data on a Serial Interface

The MCC and ATM CAS are synchronized with the superframe block boundary. At the ATM side, the structured block size should be set to the superframe block size plus the size of the CAS block so that the structured pointer inserted by the ATM controller points to the start of the structured data block. At the MCC side, the MCC is synchronized with the frame sync signal; the external framer has the ability to place the signaling information at the appropriate place in a superframe. The MPC8560 supports an automatic mode for forwarding the signaling information from the TDM to the ATM and vice versa. The ATM

controller maintains two CAS blocks per trunk (eight total). One contains the signaling information unpacked from the AAL1 cells (ATM-to-TDM), and the other contains the signaling information fetched from an external framer (TDM-to-ATM). All 8 CAS blocks reside in the internal RAM.



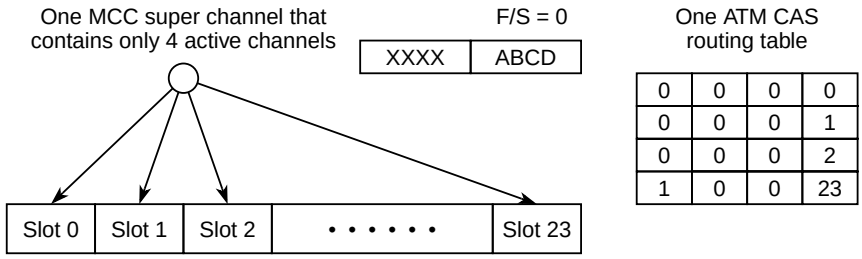
Note: CAS block resides in the internal RAM. To allow maximum flexibility in working with external framers, the signaling nibble can occupy the first or second nibble in the CAS entry.

Figure 36-8. Internal CAS Block Formats per Trunk

36.4.7 Mapping VC Signaling to CAS Blocks

Each ATM channel is connected to a specific CAS block. The ATM controller implements CAS routing tables (CRT) to maintain the routing of the signaling information from the AAL1 cells to the internal CAS blocks for receiving, and vice versa for transmitting. Each ATM channel is connected to one receive or transmit routing table. A CRT resides in a 32-byte space with each entry pointing to one signaling nibble. To allow maximum flexibility with external framers, the signaling nibble can occupy the first or second nibble in the internal CAS block (depicted in Figure 36-8).

The CRT entries should be initialized only once before the ATM channel is enabled (receiver or transmitter). The number of entries that should be initialized must be equal to the number of active slots (channels) in the corresponding MCC super channel. Each super channel is mapped to a unique ATM connection (VC). The CRT entries are in ascending order based on the channel slots assigned for the MCC super channel (depicted in Figure 36-9).



Note: Example of one MCC super channel in ESF framing (T1) containing 4 TDM slots connected to an ATM channel with one CAS routing table (CRT). (See 36.4.7.1, “CAS Routing Table.”)

Figure 36-9. Mapping CAS Entry

36.4.7.1 CAS Routing Table

Figure 36-10 shows the structure of a CAS routing table. The CP maintains a pointer which steps through the table and wraps back to the beginning of the table after servicing the last entry (W = 1).

	Byte			
First Signaling Nibble	W = 0	—	F/S	Signaling Offset Pointer
	W = 0	—	F/S	Signaling Offset Pointer
	W = 0	—	F/S	Signaling Offset Pointer
ATM Current Pointer	W = 0	—	F/S	Signaling Offset Pointer
	W = 0	—	F/S	Signaling Offset Pointer
Last Signaling Nibble	W = 1	—	F/S	Signaling Offset Pointer

Figure 36-10. AAL1 CAS Routing Table (CRT)

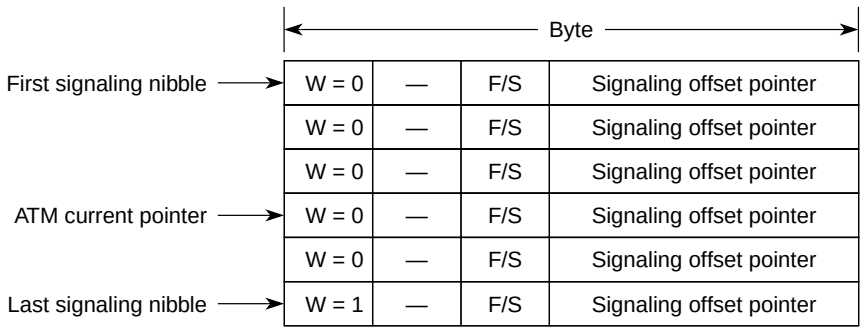


Figure 36-11 describes the structure of a CAS routing table entry.

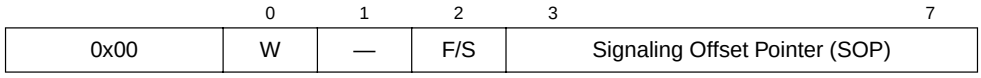


Figure 36-11. AAL1 CAS Routing Table Entry

Table 36-1 describes CAS routing table entry fields.

Table 36-1. CAS Routing Table Entry Field Descriptions

Bits	Name	Description
0	W	Wrap bit. Identifies the last entry in a circular table. During initialization, the host must clear all W bits in the table except the last one, which must be set. 0 The entry is not the last entry. The next entry is the current entry plus one. 1 Designates the last circular table entry. Causes the beginning of table to be the next entry.
1	—	Reserved, should be cleared during initialization.
2	F/S	First/Second. Specifies the location of the signaling information as either first or second nibble. 0 The signaling information occupies the first nibble in the CAS block (LSB). 1 The signaling information occupies the second nibble in the CAS block (MSB).
3-7	SOP	Signaling Offset Pointer. Offset of the signaling nibble from the internal CAS base address. Note that in ESF mode the maximum offset is 23 and in E1 framing format the maximum offset is 31.

36.4.7.2 TDM-to-ATM CAS Support

During the segmentation process, the AAL1 transmitter reads the CAS data from the internal CAS block and packs the data and the signaling information at the end of an AAL1 super frame (depicted in Figure 36-3.). All AAL1 functions operate normally (generating AAL1 PDU-headers, structured pointers, etc.). Each common (MCC, ATM) BD table should point to buffers that can contain a whole number of super frames. The last buffer of the super frame is marked as the end of a super frame (BD[EOSF] = 1). After closing a buffer with the EOSF indication, the ATM transmitter processes the CAS data—reads it from the internal CAS block and inserts it into the cell payload at the transmit side. The EOSF indication in the BD is statically set by the CPU when initializing the BD table. Figure 36-12.

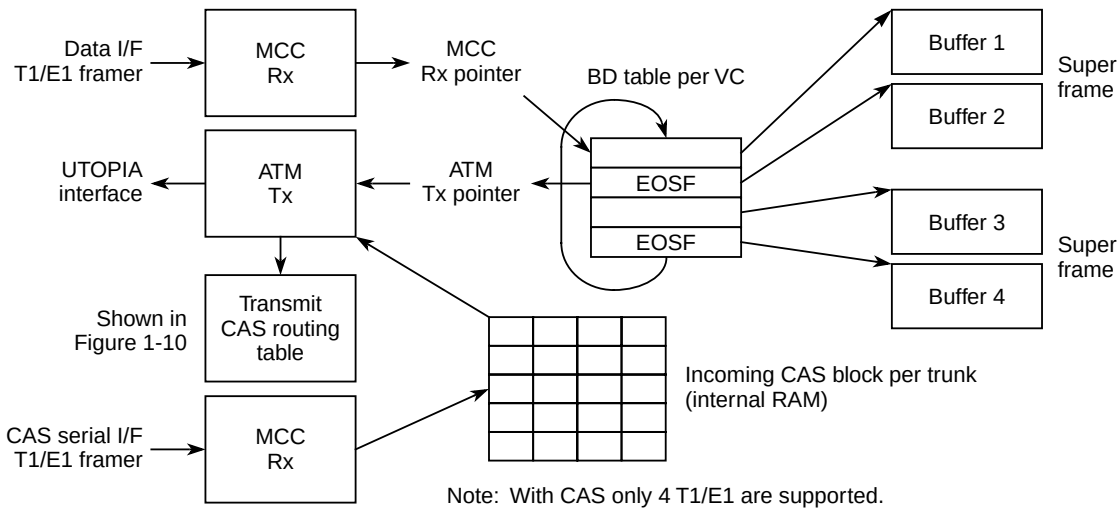


Figure 36-12. CAS Flow TDM-to-ATM

The CAS block is automatically written to internal RAM by the MCC receiver using a separate TDM. When a super frame is received the MCC should be triggered with a super-frame (multi-frame) SYNC from the external framer. The incoming CAS block should be captured by the MCC only once for each super frame (on the last frame).

External logic may be used to convert the framer super-frame SYNC to trigger the MCC. The MCC captures the CAS block from the external framer and copies it transparently into one of four internal CAS blocks. Each byte in the CAS block contains a nibble of valid CAS information (depicted in Figure 36-8).

Note that the buffer data size should not include the CAS octets.

36.4.7.2.1 CAS Mapping Using the Core (Optional)

To avoid using another TDM dedicated to CAS information, a parallel interface (controlled by the core) can be used to deliver the CAS information from the framer to the incoming CAS block. In this case, the core service routine reads the CAS information from the external framer and writes it to the incoming CAS block. To optimize the process, the framer can interrupt the core only when new information is received.

36.4.7.3 ATM-to-TDM CAS Support

The the frame reassembly proces is shown in Figure 36-13. During reassembly, the AAL1 receiver unpacks the signaling information from the end of an AAL1 super frame (shown in Figure 36-3), and places it in the internal CAS block (using the receive CAS routing table). All AAL1 functions, (such as AAL1 PDU-header verification, bit count integrity, 3-step-SN algorithm), operate normally. Each common (MCC, ATM) BD table should point to buffers that can store a whole number of super frames. The last buffer of the super frame is marked with EOSF. After closing a buffer with an EOSF indication, the ATM receiver processes the CAS data (copies it to the internal CAS block from the AAL1 cell payload at the receive side). The EOSF indication in the BD is statically set by the CPU while initializing the BD table.

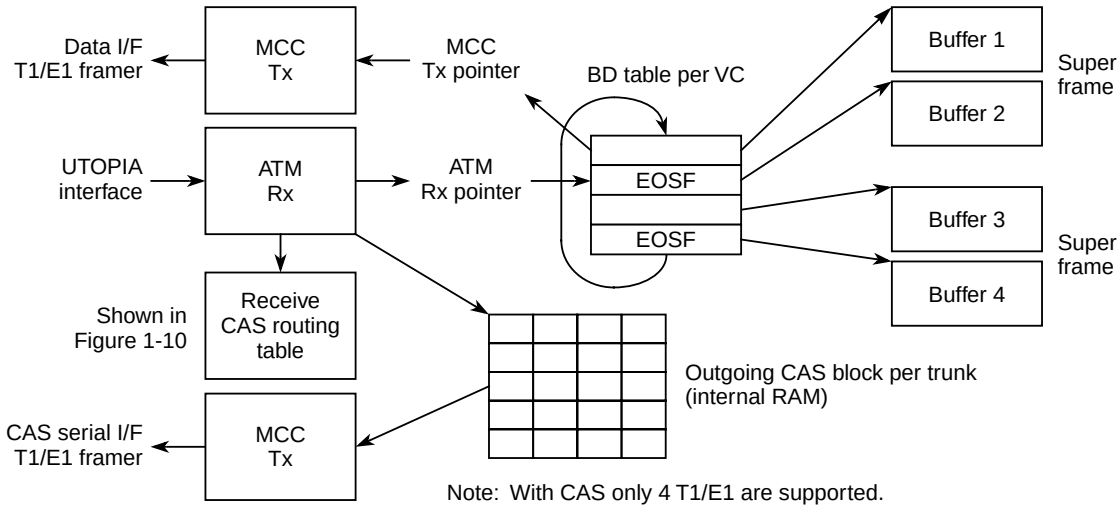


Figure 36-13. CAS Flow ATM-to-TDM

The CAS block is read from the internal RAM by the MCC. The MCC transmitter continuously reads the signaling information from one of the four outgoing internal CAS blocks and writes it transparently into the external framer. Each byte in the CAS block contains one nibble of valid CAS information (depicted in Figure 36-8).

Note that the buffer data size should not include the CAS octets.

36.4.7.3.1 CAS Updates Using the Core (Optional)

To avoid using another TDM dedicated to CAS information, the user can use a parallel interface controlled by the core to deliver the outgoing CAS block to the framer. In this case, the ATM receiver should be programmed to operate in core CAS modify mode RCT[CCASM = 1] (and RCT[CASM = 1]). In this mode, the CP adds an entry to the ATM interrupt queue and sets the appropriate sticky bit in OCASSR each time an AAL1 cell is received with new signaling information (one or more signaling nibble has changed). (See Section 36.10, “Outgoing CAS Status Register (OCASSR).”) A core interrupt service routine can then write the updated outgoing CAS block to the framer.

Note to avoid additional latency, the interrupt queue assigned to this connection should have a global interrupt threshold of one. See the INT_ICNT parameter discussed in section 29.11.3 of the ATM chapter.

36.5 ATM-to-TDM Adaptive Slip Control

Two types of slip can occur in ATM-to-TDM operation: overrun and underrun. The two cases are handled by the MCC and ATM controller automatically without requiring CPU intervention.

Overrun occurs when the MCC transmitter fetches data from the common BD table at a slower rate than it is being filled by the ATM receiver (shown in Figure 36-15.). In this case, the ATM write pointer meets the MCC read pointer and a BSY state is declared (an entry is added to the ATM interrupt table) on the ATM side.

Underrun occurs when the MCC transmitter fetches data from the common BD table at a higher rate than it is being filled by the ATM receiver (shown in). In this case, the MCC read pointer reaches a BD that is not ready and a buffer-not-ready state is declared on the MCC side.

In both slip cases, the MCC and ATM controller automatically recover and restart the ATM-to-TDM interworking function.

In order to prevent overrun and underrun conditions, the MPC8560 maintains an adaptive slip control using a set of four threshold pointers for each ATM-TDM (VC - super channel) connection.

36.5.1 Pre-Underrun

The pre-underrun state (shown in Figure 36-14) occurs when the MCC read pointer advances faster than the ATM write pointer. When the adaptive counter reaches the MCC_Stop threshold, the ATM receiver continues to accept valid data and advance its write pointer, but the MCC read pointer does not advance. Instead, the current buffer (or the underrun template) is retransmitted.The number of retransmissions is an integer multiple of the frame size. The MCC resumes normal data transmission when the adaptive counter reaches the MCC_start threshold. After completing its current buffer retransmission, (the one in progress when the start threshold was reached), the MCC begins advancing its read pointer and transmitting the newly received data.

Note that when the pre-underrun state occurs, the MCC can continuously transmit either the last buffer, or the underrun template, as determined by CHAMR[UTM]. See Section 36.18.2, “Channel Mode Register (CHAMR)—CES Mode,” for more MCC channel configuration information. Figure 36-14 shows the MCC retransmitting the current buffer.

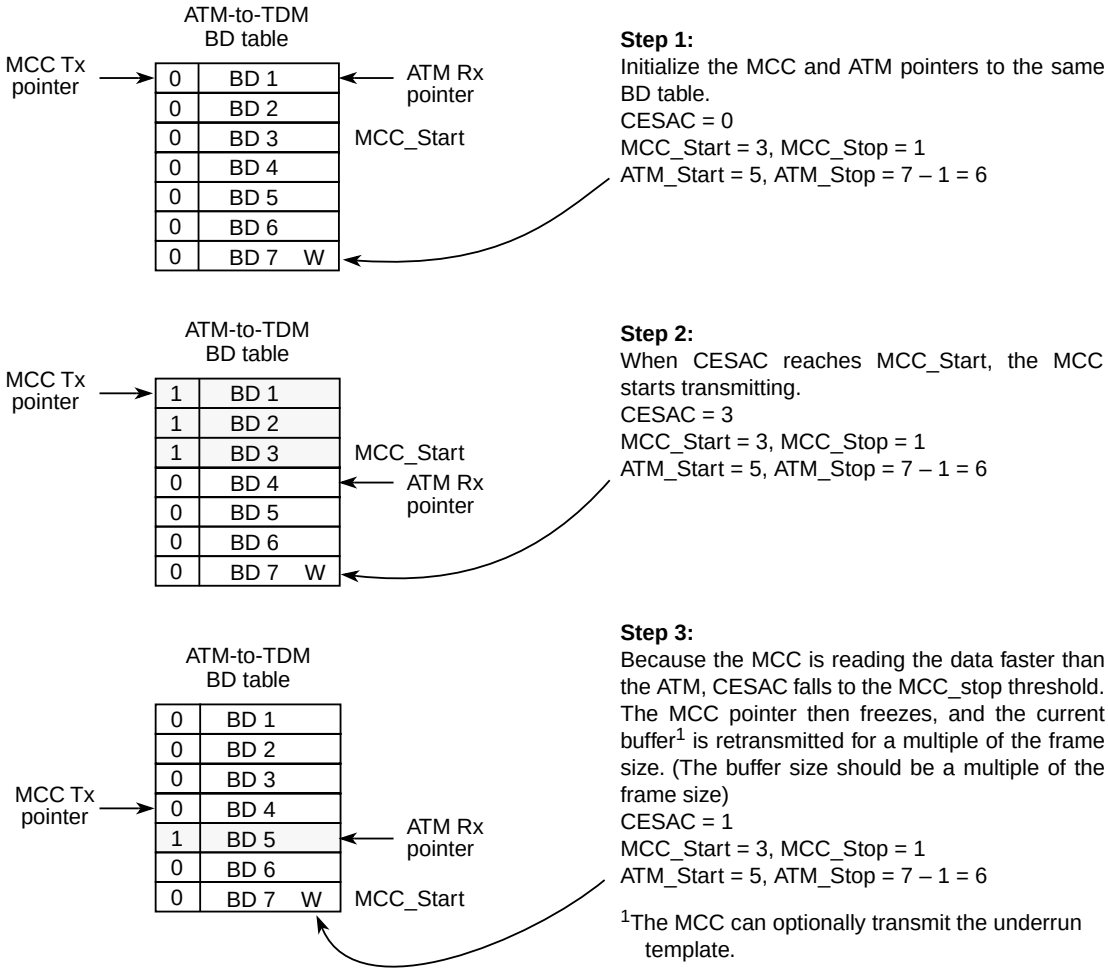


Figure 36-14. Pre-Underrun Sequence

36.5.2 Pre-Overrun

The pre-overrun state,, shown in Figure 36-15, occurs when the ATM write pointer advances faster than the MCC read pointer. When the adaptive counter reaches the ATM_Stop threshold, the MCC read pointer continues to process valid data from the common BD table, but the ATM write pointer does not advance. Instead, the ATM receiver waits until the adaptive counter reaches the ATM_start threshold.

When the CESAC reaches the ATM start threshold, the ATM write pointer advances to the first BD after the one that marked with EOSF (in CAS mode). When this BD is ready, the ATM receiver begins the resynchronization process:

- for unstructured AAL1 type the ATM receiver waits for the first valid cell,
- for structured AAL1 type the receiver waits for the first valid cell that contains a valid pointer. The first received octet becomes the first byte of the new BD (new super frame).

Note that this implementation of slip control provides a good interface for an adaptive FIFO implemented in software. CESAC already represents the difference between the ATM and MCC pointers; the application software need only convert this value into an SRTS format.

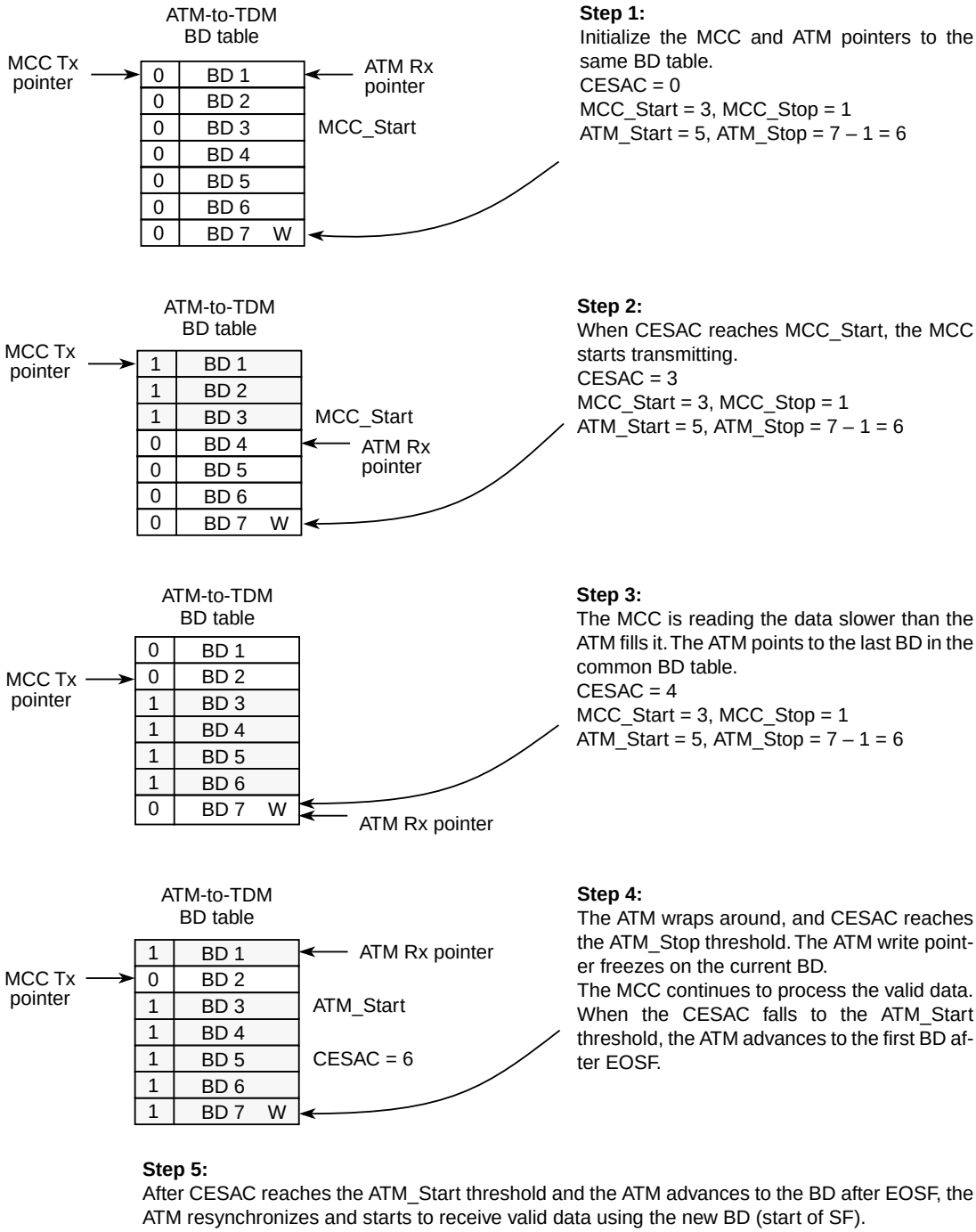
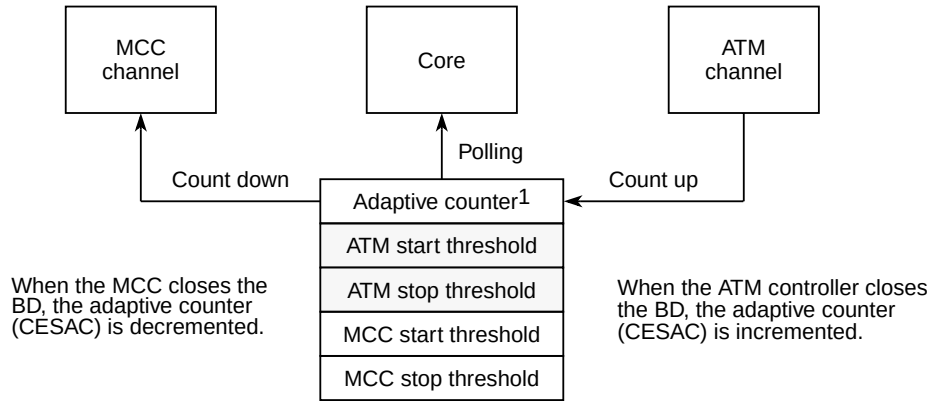


Figure 36-15. Pre-Overrun Sequence

Figure 36-16 shows the 8-byte data structure used to implement ATM-to-TDM slip control. (Three of the bytes are unused.) Note the MCC start threshold, in effect, implements a CDV

jitter buffer. The MCC and ATM stop thresholds determine the CDVT. In the following list, *b* and *a* represent integers less the BD table size.

- MCC stop threshold should be programmed to (BD Table size – *b*) to prevent buffer underrun.
- ATM stop threshold should be programmed to (BD Table size – *a*) to prevent buffer overrun.
- The ATM start threshold determines the amount of time the ATM receiver waits before restarting the synchronization process.
- (MCC start – MCC stop) > = frame size.
- The CES adaptive threshold table address is given by:
  (CATB + MCC_Super_Channel) x 8



1 CES adaptive threshold table address: (CATB + MCC_Super_Channel) x 8

Figure 36-16. Data Structure for ATM-to-TDM Adaptive Slip Control

36.5.3 CES Adaptive Threshold Tables

The CES adaptive threshold tables (see Table 36-2) reside in the dual-port RAM and hold the CES thresholds on a per-VC basis. The CES adaptive threshold base (CATB), located in the AAL1 parameter RAM, points to the base address of these tables. Each AAL1-MCC channel has its own table with a starting address given by CATB + RCT[Super_Channel_Number]# × 8.

Local Offset	0 7	8 15
0x00	—	CES adaptive counter
0x02	ATM stop threshold	ATM start threshold
0x04	MCC stop threshold	MCC start threshold
0x06	—	

Table 36-2. CES Adaptive Threshold Table

Table 36-3 describes CES adaptive threshold table fields.

Table 36-3. CES Adaptive Threshold Table Field Descriptions

Offset [1]	Bits	Name	Description
0x00	0–7	—	Reserved, should be cleared during initialization.
	8–15	CESAC	CES adaptive counter.This field contains the difference between the ATM write pointer and the MCC read pointer on the common BD table. Should be cleared by the user during the channel initialization.
0x02	0–7	ASTP	ATM stop threshold.This threshold determines the maximum amount of buffering (CDVT) that required in the common BD table (shown in Figure 36-15.). When the CESAC reaches this value a pre-overrun condition occurs. Should be defined by the user during the channel initialization.
	8–15	ASTRT	ATM start threshold. Determines the time interval the ATM controller takes to restart the synchronize process (shown in Figure 36-15.) after pre-overrun condition. Should be defined by the user during the channel initialization.
0x04	0–7	MSTP	MCC stop threshold.This threshold determines the minimum amount of buffering in the common BD table (shown in).When the CESAC reaches this value a pre-underrun condition occur and the MCC starts to transmit the underrun template or the last BD (see 36.5, “ATM-to-TDM Adaptive Slip Control”). Should be defined by the user during the channel initialization.
	8–15	MSTRT	MCC start threshold.This threshold determines the effective depth of the jitter buffer, the amount of buffering required to cope with the ATM network’s CDV (shown in).When the CESAC reaches this value the MCC transmitter starts sending the valid data from the common BD table. Should be defined by the user during the channel initialization.
0x06	0–15	—	Reserved, should be cleared during initialization.

[1] The offset is CATB + RCT[Super_Channel_Number]# × 8

36.6 3-Step-SN Algorithm

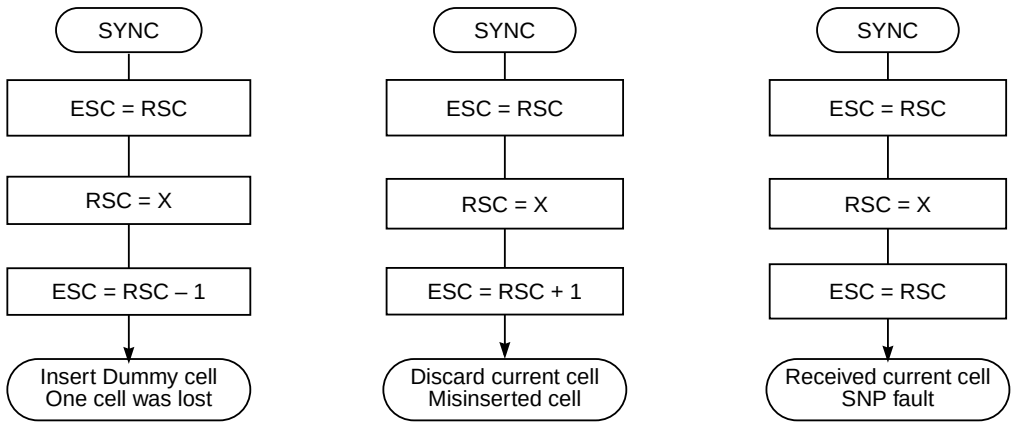
The 3-step-SN algorithm is a fast and efficient state machine that has the ability to recover one lost or misinserted cell. The 3-step-SN algorithm doesn’t add significant delay to the AAL1 reassembly process due to the fact that the decision to accept a cell is taken immediately after cell arrival. If a lost cell is detected, the receiver will insert a dummy cell. If a misinserted cell is detected, the receiver will discard the cell received after the misinserted cell. If more than one successive cell was lost or misinserted, the 3-step-SN

algorithm will switch to hunt mode, where it stays until a cell with a valid SN field is received.

36.6.1 The Three States of the Algorithm

The 3-step-SN algorithm has three states:

1. Hunt—The Hunt state is the initial state of the algorithm or the first state after the receiver loses its synchronization. In this state no cell is delivered to the Rx buffer. When a valid cell is detected, the algorithm switches to the Sync state and delivers the current cell to the Rx buffer.
2. Sync—The Sync state is the steady state of the algorithm. In this state any received cell is automatically passed to the Rx buffer. The algorithm will switch from this state when an invalid cell is received (SNP error), or it receives a cell that does not sequence with the last valid cell (SC error).
3. Sync Fail—The Sync Fail state is a transient state. In this state the receiver checks the difference between the received sequence count (RSC) and the expected sequence count (ESC). If the difference between the two (ESC-RSC) is -1, 0, or 1, the receiver switches to sync mode and either discards the current cell, receives the current cell, or inserts a dummy cell, correspondingly. In any other cases the receiver will switch to hunt mode and discard the current cell.



ESC: Expected Sequence Count.
RSC: Received sequence count.
RSN = X: invalid cell received or, a cell that does not match to the ESC.

Figure 36-17. Recoverable Sync Fail Sequence Options

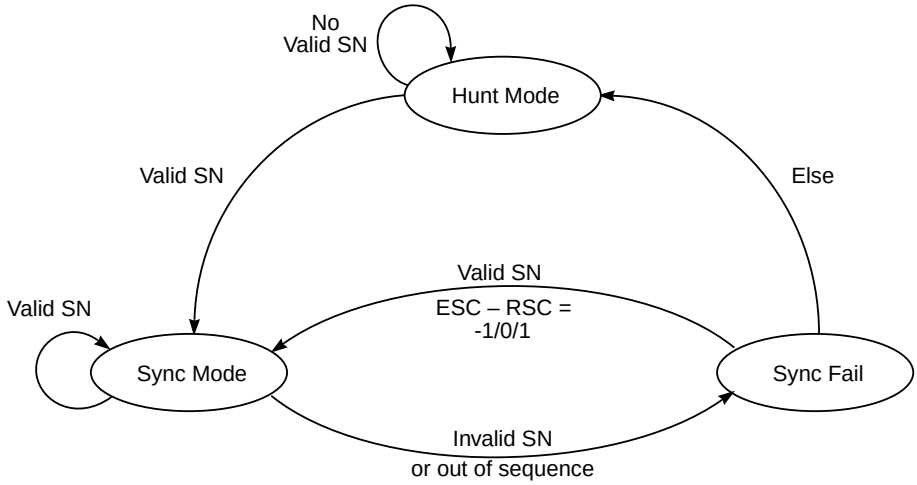


Figure 36-18. 3-Step-SN-Algorithm

36.7 Pointer Verification Mechanism

After the 3-step-SN algorithm processes the incoming cells, the cells status (Valid, Tag, Drop or Dummy) is delivered to the pointer verification mechanism. This state machine calculates the expected received pointer. If the current cell is valid and supposed to deliver a pointer, the received pointer is compared with the expected one. In the case of pointer mismatch or pointer error (for example, a parity error), the pointer state machine switches to pre-hunt mode. If the cell is not valid (for example tag, drop, or dummy) and is supposed to carry a pointer, the pointer is declared a mismatch and the pointer state machine switches to pre-hunt mode. The receiver stays in this transient state for only one AAL1 cell cycle (eight successive cells). When the pointer received in this state is different from the expected one, or had a non-valid status, the receiver switches to hunt mode where it stays until a valid cell with a “start” structured pointer is received to regain synchronization.

Note that a “start” pointer is a valid pointer with a value not equal to 93 or 127.

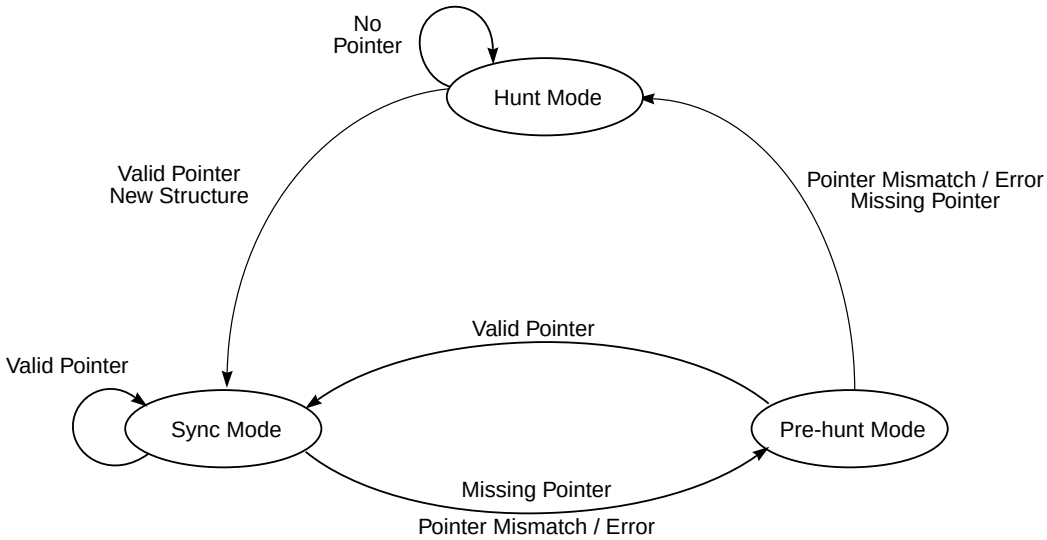


Figure 36-19. Pointer verification mechanism

36.8 AAL-1 Memory Structure

The CPM manages ATM traffic by means of transmit and receive buffer descriptors (BD) and by transmit and receive connection tables (referred to as TCTs and RCTs, respectively). Buffer descriptors are circular lists of pointers into transmit and receive buffer space in external memory. The following sections describe the organization and configuration of the buffer descriptors, TCTs, and RCTs.

36.8.1 AAL1 Parameter RAM

The MPC8560 parameter RAM is used to configure the three FCCs. The CP also uses the parameter RAM to store operational and temporary values. When configured for ATM mode, MPC8560 overlays the configuration structures shown in the next tables. The following table includes fields for ATM operation generally and AAL1 fields particularly.

Note that some of the values must be initialized by the user; values set by the CP should not be modified by the user.

Table 36-4. AAL1 Field Descriptions

Offset	Name	Width	Description
0x00–0x3F	—	—	Reserved, should be cleared during initialization..
0x40	RCELL_TMP_BASE	Hword	Rx cell temporary base address. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64 byte aligned. User-defined. (recommended address space: 3000h-4000h or b000h-c000h)

Table 36-4. AAL1 Field Descriptions (continued)

Offset	Name	Width	Description
0x42	TCELL_TMP_BASE	Hword	Tx cell temporary base address. Points to total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User-defined. (recommended address space: 3000h-4000h or b000h-c000h)
0x44	UDC_TMP_BASE	Hword	UDC mode only. Points to a total of 32 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User-defined. (recommended address space: 3000h-4000h or b000h-c000h)
0x46	INT_RCT_BASE	Hword	Internal receive connection table base. User-defined.
0x48	INT_TCT_BASE	Hword	Internal transmit connection table base. User-defined.
0x4A	INT_TCTE_BASE	Hword	Internal transmit connection table extension base. User-defined. Note that the TCT extension is not needed in AAL1 and that the AAL1 microcode uses this Hword to point to a CAS routing table.
0x4C	—	Word	Reserved, should be cleared during initialization.
0x50	EXT_RCT_BASE	Word	External receive connection table base. User-defined.
0x54	EXT_TCT_BASE	Word	External transmit connection table base. User-defined.
0x58	EXT_TCTE_BASE	Word	External transmit connection table extension base. User-defined. Note that the TCT extension is not needed in AAL1 and that the AAL1 microcode uses this Hword to point to a CAS routing table.
0x5C	UEAD_OFFSET	Hword	User-defined cells mode only. The offset of the UEAD entry in the UDC extra header. Should be an even address. If RCT[BO] = 01 UEAD_OFFSET should be in little-endian format. For example if UEAD entry is the first half word of the extra header in external memory, UEAD_OFFSET should be set to 2 (second half word entry in internal RAM).
0x5E	—	Hword	Reserved, should be cleared during initialization.
0x60	PMT_BASE	Hword	Performance monitoring table base. User-defined.
0x62	APCP_BASE	Hword	APC parameters table base address. User-defined.
0x64	FBT_BASE	Hword	Free buffer pool parameters table base. User-defined.
0x66	INTT_BASE	Hword	Interrupt queue parameters table base. User-defined.
0x5E	—	—	Reserved, should be cleared during initialization.
0x6A	UNI_STATT_BASE	Hword	UNI statistics table base. User-defined.
0x6C	BD_BASE_EXT	Word	BD table base address extension. BD_BASE_EXT[0–7] holds the 8 left bits of the Rx/Tx BD table base address. BD_BASE_EXT[8–31] should be zero. User-defined.
0x70	VPT_BASE / EXT_CAM_BASE	Word	Base address of the address compression VP table/external CAM. User-defined.
0x74	VCT_BASE	Word	Base address of the address compression VC table. User-defined.
0x78	VPT1_BASE / EXT_CAM1_BASE	Word	Base address of the address compression VP1 table/EXT CAM1. User-defined.
0x7C	VCT1_BASE	Word	Base address of the address compression VC1 table. User-defined.

Table 36-4. AAL1 Field Descriptions (continued)

Offset	Name	Width	Description
0x80	VP_MASK	Hword	VP mask for address compression lookup. User-defined.
0x82	VCI_Filtering	Hword	VCI filtering enable bits. When cells with VCI = 3, 4, 6, 7–15 are received and the associated VCI_Filtering bit = 1 the cell is sent to the raw cell queue. VCI = 3 is associated with VCI_Filtering[3], VCI = 15 is associated with VCI_Filtering[15]. VCI_Filtering[0–2, 5] should be zero. See Section 29.10.1.2.
0x84	GMODE	Hword	Global mode. User-defined. See Section 29.10.1.3, “Global Mode Entry (GMODE).”
0x86	COMM_INFO	Hword	The information field associated with the last host command. User-defined. See Section 29.14, “ATM Transmit Command.”
0x88		Hword	
0x8A		Hword	
0x8C	Reserved	Word	Reserved, should be cleared during initialization.
0x90	CRC32_PRES	Word	Preset for CRC32. Initialize to 0xFFFFFFFF.
0x94	CRC32_MASK	Word	Constant mask for CRC32. Initialize to 0xDEBB20E3.
0x98	AAL1_SNPT_BASE	Hword	AAL1 SNP protection look up table base address. (AAL1 only.) The 32-byte table resides in dual-port RAM and must be initialized by the user (See Section 36.14, “AAL1 Sequence Number (SN) Protection Table.”).
0x9A	—	Hword	Reserved, should be cleared during initialization.
0x9C	SRTS_BASE	Word	External SRTS logic base address. (AAL1 only.) Should be 16-byte aligned.
0xA0	IDLE/UNASSIGN_BASE	Hword	Idle/unassign cell base address. Points to dual-port RAM area contains idle/unassign cell template (little-endian format). Should be 64-byte aligned. User-defined. The ATM header should be 0x0000_0000 or 0x0100_0000 (CLP = 1).
0xA2	IDLE/UNASSIGN_SIZE	Hword	Idle/Unassign cell size. 52 in regular mode. 53–64 in UDC mode.
0xA4	EPAYLOAD	Word	Reserved payload. Initialize to 0x6A6A6A6A.
0xA8	Trm	Word	(ABR only) The upper bound on the time between F-RM cells for an active source. TM 4.0 defines the Trm period as 100 msec. The Trm value is defined by the system clock and the time stamp timer prescaler (See RTSCR). For time stamp prescalar of 1µS, Trm should be set to 100 ms/1µs = 100,000.
0xAC	Nrm	Hword	(ABR only) Controls the maximum cells the source may send for each F-RM cell. Set to 32 cells.
0xAE	Mrm	Hword	(ABR only) Controls the bandwidth between F-RM, B-RM and user data cell. Set to 2 cells.
0xB0	TCR	Hword	(ABR only) Tag cell rate. The minimum cell rate allowed for all ABR channels. An ABR channel whose ACR is less then TCR sends only out-of-rate F-RM cells at TCR. Should be set to 10 cells/sec as defined in the TM 4.0. Uses the ATMF TM 4.0 floating-point format. Note that the APC minimum cell rate should be at least TCR.
0xB2	ABR_RX_TCTE	Hword	(ABR only) Points to total of 16 bytes reserved dual-port RAM area used by the CP. Should be aligned on 16-byte boundary. User-defined.

Additional CES parameters needed by the AAL1 microcode are described in the following table.

Table 36-5. AAL1 CES Parameters

Offset	Name	Width	Description
0x4A	INT_RTCRT_BASE	Hword	Internal receive/transmit CAS routing table extension base. User-defined. Note that because AAL1 does not need the TCT extension, the AAL1 microcode uses this Hword to point to a CAS routing table. Should be aligned on 32-byte boundary. User-defined.
0x58	EXT_RTCRT_BASE	Word	External receive/transmit CAS routing table extension base. User-defined. Note that because AAL1 does not need the TCT extension, the AAL1 microcode uses this word to point to a CAS routing table.
0xB2	AAL1_INT_RX_CRT	Hword	(CES only) Points to a reserved scratchpad area of 32 bytes in the dual-port RAM used by the CES microcode. Should be aligned on 32-byte boundary. User-defined.
0xd0	OCASSR	Byte	Outgoing CAS Status Register. See Section 36.10, “Outgoing CAS Status Register (OCASSR).”
0xE0	TCELL_TMP_BASE_EXT	Word	Transmit Cell Temporary base address (64-byte aligned). Points to an external memory block reserved for partially filled cells (64 octets for each CES channel). This area is allocated by the user but used by the CP.
0xE4	IN_CAS_BLOCK_BASE	Hword	Incoming CAS Block Base . Points to dual-port RAM. Should be 32-byte aligned. User-defined.
0xE6	OUT_CAS_BLOCK_BASE	Hword	Outgoing CAS Block Base (depicted in Figure 36-10). Points to dual-port RAM. Should be 32-byte aligned. User-defined.
0xE8	AAL1_Int_STATT_BASE	Hword	AAL1 Internal Statistics Table Base. Points to dual-port RAM. Should be 16-byte aligned. User-defined. See 36.15, “Internal AAL1 Statistics Tables.”
0xEA	AAL1_DUMMY_CELL_BASE	Hword	ALL1 Dummy cell base address. Points to dual-port RAM area contains the AAL1 dummy cell template (little-endian format). Should be 64-byte aligned. User-defined.
0xEC	CATB	Hword	CES adaptive threshold tables base address. Points to the dual-port RAM area containing the CES slip control thresholds and the adaptive counter See 36.5, “ATM-to-TDM Adaptive Slip Control.” Should be 8-byte aligned (8 octets for each AAL1-MCC channel). User-defined and should match the CATB value programmed in the MCC parameter RAM; see Section 36.18.1, “CES Additions to the MCC Parameter RAM.”
0xF0	AAL1_Ext_STATT_BASE	Word	AAL1 External Statistics Table Base. Points to External memory. Should be 16-byte aligned (16 octets for each AAL1 channel).User-defined. See Section 36.16, “External AAL1 Statistics Tables.”

36.9 Receive and Transmit Connection Tables (RCT, TCT)

The connection tables, RCT and TCT, hold channel configuration and temporary parameters for each receive and transmit channel (AAL type, connection traffic parameters, BDs' parameters and temporary parameters used during segmentation and reassembly).

The internal connection tables hold parameters for up to 128 channels (channels 0–127). In extended channel mode, parameters for channels 256 and above are kept in external memory. The only relationship between transmit and receive connection tables of the same channel is the CRT (CAS routing table); see 36.4.7.1, "CAS Routing Table."

Each connection table entry resides in 32 bytes. The pointers to these connection tables reside in the parameter RAM.

36.9.1 Receive Connection Table (RCT)

Figure 36-20 shows the format of an RCT entry.

Local Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x00	—		GBL	BO		—	DTB	BIB	—		SEGF	ENDF	—	SYNC	INTQ	
0x02	—	—	—									CASM	ABRF	AAL		
0x04	RX Data Buffer Pointer (RXDBPTR)															
0x06																
0x08	Cell Time Stamp															
0x0A																
0x0C	RBD_Offset															
0x0E–0x18	Protocol Specific															
0x1A	MRBLR															
0x1C	—		PMT						RBD_BASE							
0x1E	RBD_BASE											CCASM	CESM	—	PM	

Figure 36-20. Receive Connection Table (RCT) Entry

NOTE

For an active channel, the CP uses a burst cycle to fetch the 32-byte RCT and writes back only the first 24 bytes.

Table 36-6 describes RCT fields.

Table 36-6. RCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared during initialization.
	2	GBL	Global. Asserting GBL enables snooping of data buffers, BD, interrupt queues and free buffer pool.
	3–4	BO	Byte ordering—used for data buffers. 0x Reserved 1x Big endian
	5	—	Reserved, should be cleared during initialization.
	6	DTB	Data buffers bus 0 Data buffers reside on the system bus. 1 Data buffers reside on the local bus.
	7	BIB	BD interrupt queues, free buffer pool, and CES external statistics tables bus placement 0 Reside on the system bus. 1 Reside on the local bus. Notes: • When in UDC mode (AAL5 or AAL1), BDs and data should be placed on the same bus (RCT[DTB] = RCT[BIB]). • RCT[BIB] programming must be consistent across all CES receive channels because they share the same AAL1_Ext_STATT_BASE parameter.
	8–9	—	Reserved, should be cleared during initialization.
	10	SEGF	OAM F5 segment filtering 0 Do not send cells with PTI = 100 to the raw cell queue. 1 Send cells with PTI = 100 to the raw cell queue.
	11	ENDF	OAM F5 end-to-end filtering 0 Do not send cells with PTI = 101 to the raw cell queue. 1 Send cells with PTI = 101 to the raw cell queue.
	12	—	Reserved, should be cleared during initialization.
	13	SYNC	AAL1 SYNC. The user should set this bit for first AAL1 synchronization. Used internally by the CP.
	14–15	INTQ	Points to one of four interrupt queues available. 00 Interrupt queue 0 selected 01 Interrupt queue 1 selected 10 Interrupt queue 2 selected 11 Interrupt queue 3 selected

Table 36-6. RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	0–3	—	Internal use only. Initialize to 0.
	4–10	—	Internal use only. Initialize to 0.
	11	CASM	Common associated signaling mode. 0 CAS operation mode is disable. 1 CAS operation mode is enable.
	12	—	Reserved, should be cleared during initialization.
	13–15	AAL	AAL type 000 AAL0 —Reassembly with no adaptation layer 001 AAL1 —ATM adaptation layer 1 010 AAL5 —ATM adaptation layer 5 011 AAL3/4 —ATM adaptation layer 3/4 100 AAL2 —ATM adaptation layer 2 101 AAL1_CES —ATM adaptation layer 1 with circuit emulation service All others reserved.
0x04	—	RxDBPTR	Receive data buffer pointer. Holds real address of current position in the Rx buffer.
0x08	—	Cell Time Stamp	Used for reassembly time-out. Whenever a cell is received, the MPC8560 time stamp timer is sampled and written to this field. See the RISC Time-Stamp Control Register (RTSCR) discussion in the CPM chapter.
0x0C	—	RBD_Offset	RxBD offset from RBD_BASE. Points to the channel's current BD. User-initialized to 0; updated by the CP.
0x0E–0x18		—	Protocol-specific area.
0x1A	—	MRBLR	Maximum receive buffer length. Used in both static and dynamic buffer allocation. Note that in CES mode (CESM = 1) this value must be a multiple of eight (MCC limited).

Table 36-6. RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1C	0–1	—	Reserved, should be cleared during initialization.
	2–7	PMT	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE+PMT × 32. Can be changed on-the-fly.
	8–15	RBD_BASE	RxBD base. Points to the first BD in the channel's RxBD table. The 8 most-significant bits of the address are taken from BD_BASE_EXT in the parameter RAM. The four least-significant bits of the address are taken as zeros.
0x1E	0–11		
	12	CCASM	Core CAS modify. When this mode is enabled, the CP sets OCASSR[MCASB*n*] sticky bit each time the outgoing (ATM to TDM) CAS block is changed. 0 Core CAS modify mode is disabled. 1 Core CAS modify mode is enabled. See Section 36.10, "Outgoing CAS Status Register (OCASSR)."
	13	CESM	Circuit Emulation Service Mode. 0 CES operation mode is disable. Adaptive Slip control mechanism is disabled. 1 CES operation mode is enable. Adaptive Slip control mechanism is enabled.
	14	—	Reserved, should be cleared during initialization.
	15	PM	Performance monitoring. Can be changed on-the-fly. 0 No performance monitoring for this VC. 1 Perform performance monitoring for this VC. Whenever a cell is received for this VC the performance monitoring table that its code is written in the PMT field is updated.

36.9.1.1 AAL1 Protocol-Specific RCT

Figure 36-21 shows the AAL1 protocol-specific area of an RCT entry.

Local Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0E	SRTS_DEV				—				PFM	SRT	INVE	STF	—			
0x10	OCASB/SRTS_TMP				—				—		Valid Octet Size (VOS)					
0x12	SPV	—			Structured Pointer (SP)											
0x14	RBDCNT															
0x16	Block Size												—	SN		
0x18	Super Channel Number								RXBM	SLIPIM	—		CASBS			

Figure 36-21. AAL1 Protocol-Specific RCT

Table 36-7 describes AAL1 protocol-specific RCT fields.

Table 36-7. AAL1 Protocol-Specific RCT Field Descriptions

Offset	Bits	Name	Description
0x0E	0–3	SRTS_DEV	Selects an SRTS device, whose address is SRTS_BASE[0–27] + SRTS_DEV[28–31]. The 16 byte-aligned SRTS_BASE is taken from the parameter RAM.
	4–7	—	Reserved, should be cleared during initialization.
	8	PFM	Partially filled mode. 0 Partially filled cells mode is not used. 1 Partially filled cells mode is used. The receiver copies only valid octets from the AAL1 cell to the Rx buffer. The number of the valid octets from the beginning of the AAL1 user data field is specified in the VOS (valid octet size) field.
	9	SRT	Synchronous residual time stamp. Unstructured format only. The MPC8560 supports clock recovery using an external SRTS PLL. The MPC8560 tracks the SRTS from the incoming four cells with SN = 1, 3, 5, and 7 and writes it to the external SRTS device. Every eight cells the CP writes a valid SRTS to external logic. (See Section 36.17, “SRTS Generation and Clock Recovery Using External SRTS Logic.”) 0 SRTS mode is not used. 1 SRTS mode is used.
	10	INVE	Inverted empty. 0 RxBD[E] is interpreted normally (1 = empty, 0 = not empty). 1 RxBD[E] is handled in negative logic (0 = empty, 1 = not empty). Note that in CAS mode (CESM = 1) this bit must be set by the user; see Section 36.4.1, “Automatic Data Forwarding.”
	11	STF	Structured format. 0 Unstructured format is used. 1 Structured format is used. Note that although the structured format may be used, if the block size is one, the user should clear STF. Only non P-format AAL1 cells are received. The receiver does not check the AAL1 structured pointer because there is no need to when the block size is one.
	12–15	—	Reserved, should be cleared during initialization.
0x10	0–3	OCASB/SRTS_TMP	OCASB applies when in CAS mode. Outgoing CAS Block. Points to one of the eight available internal CAS blocks. The starting address of the table is OUT_CAS_BLOCK_BASE + OCASB× 32. See 36.4.7.1, “CAS Routing Table” for more details. Note that the RCT and TCT use the same CAS routing table (CRT). SRTS_TMP applies when not in CAS mode. Used by the CP to store the received SRTS code. After a cell with SN = 7 is received, the CP writes the SRTS code to the external SRTS device. Note that when the receiver is in hunt mode SRTS information is not updated.
	4–9	—	Reserved, should be cleared during initialization.
	10–15	VOS	Valid octet size. Specifies the number of valid octets from the beginning of the AAL1 user data field. For unstructured, service values 1–47 are valid; for structured service, values 1-46 are valid. Partially filled cell mode only. 000000 no valid octets 000001 one valid octet • • • 101110 46 valid octets, upper limit for structured service. 101111 47 valid octets, upper limit for unstructured service

Table 36-7. AAL1 Protocol-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x12	0	SPV	Structured pointer valid. Should be user-initialized user to zero . Structured format only.
	1–3	—	Reserved, should be cleared during initialization.
	4–15	SP	Structured pointer. Used by the CP to calculate the structured pointer. Should be initialized by the user to zero. Used in structured format only.
0x14	—	RBDCNT	RxBD count. Indicates how many bytes remain in the current Rx buffer. Initialized with MRBLR whenever the CP opens a new buffer.
0x16	0–11	Block Size	Used only in structured format. Specifies the structured block size (Block Size = 0xFFF = 4 Kbytes maximum). Note that when working in CAS mode (CESM = 1 and CASM = 1), the block size should be programmed to the superframe block size plus the size of the CAS block. However, when working in basic mode (CES without CAS: CESM = 1 and CASM = 0) the block size should be programmed to the number of MCC slots (N x 64) per frame assigned to this channel.
	12	—	Reserved, should be cleared during initialization.
	13–15	SN	Sequence number. Used by the CP to check incoming cell's sequence number.
0x18	0–7	SCN	Super Channel Number. Should contain the MCC Super Channel Number that mapped to this ATM channel. This field must be initialized by the user in CES mode only. See Section 36.4.7, "Mapping VC Signaling to CAS Blocks."
	8	RXBM	Receive buffer interrupt mask 0 The receive buffer event of this channel is disabled. (The event is not sent to the interrupt queue.) 1 The receive buffer event of this channel is enabled.
	9	SLIPIM	Slip interrupt mask 0 The slip interrupts SLIPS and SLIPE are both masked. 1 When the receiver switches to hunt mode due to a 3-step-SN algorithm fault, or due to two successive mismatched pointers, or due to a pre-overrun condition, the SLIPS interrupt is sent to the interrupt queue. See Section Figure 36-15., "Pre-Overrun Sequence," and Section 36.13, "AAL1 Exceptions." When the receiver resynchronizes, SLIPE interrupt is sent to the interrupt queue. Note that this is the error reporting mechanism during automatic data forwarding (ATM-to-TDM bridging).
	10–11	—	Reserved, should be cleared during initialization.
	12–15	CASBS	CAS block size.This field contains the number divided by two (**N/2**) of signaling nibbles mapped to this ATM channel. See Section 36.4.7, "Mapping VC Signaling to CAS Blocks." **Note** that if the number of signaling nibbles is **odd**, this field should be programmed to (**N + 1)/2**. See Section 36.2.2, "Signaling Path." **Example**: ESF with three T1 time slots connected to one ATM channel. The **Data_Size =** 3*24 = 72 octets and the **CAS Block = 2** octets (**N = 3, (N + 1)/2 = 2**), so in this case the **Block_Size** = 72 + 2 = 74

36.9.2 Transmit Connection Table (TCT)

Figure 36-22 shows the format of an TCT entry.

Local Offset	0 1	2	3 4	5	6	7	8 9	10 11	12	13	14 15
0x00	—	GBL	BO	—	DTB	BIB	—	ATT	AVCF	VCON	INTQ
0x02	—										AAL
0x04–0x06	Tx Data Buffer Pointer (TXDBPTR)										
0x08	TBDCNT										
0x0A	TBD_OFFSET										
0x0C	Rate Remainder						PCR Fraction				
0x0E	PCR										
0x10–0x14	Protocol Specific										
0x16	APC Linked Channel										
0x18	ATM Cell Header (VPI,VCI,PTI,CLP)										
0x1a											
0x1C	—	PMT					TBD_BASE				
0x1E	TBD_BASE								BNM	STPT	IMK PM

Figure 36-22. Transmit Connection Table (TCT) Entry

Table 36-8 describes general TCT fields.

Table 36-8. TCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared during initialization.
	2	GBL	Global. Asserting GBL enables snooping of data buffers, BDs, interrupt queues and free buffer pool.
	3–4	BO	Byte ordering. This field is used for data buffers. 0x Reserved. 1x Big endian.
	5	—	Reserved, should be cleared during initialization.
	6	DTB	Data buffer and CES transmit cell scratchpad bus placement 0 Reside on the system bus. 1 Reside on the local bus. Note: TCT[DTB] programming must be consistent across all CES transmit channels because they share the same TCELL_TMP_BASE_EXT parameter.
	7	BIB	BD and interrupt queue bus 0 Reside on the system bus. 1 Reside on the local bus. Note: When using AAL5, AAL1 in UDC mode, BDs and data should be placed on the same bus (TCT[DTB] = TCT[BIB]).
	8–9	—	Reserved, should be cleared during initialization.
	10–11	ATT	ATM traffic type 00 Peak cell-rate pacing. The host must initialize PCR and the PCR fraction. Other traffic parameters are not used. 01 Peak and sustain cell rate pacing (VBR traffic). The APC performs a continuous-state leaky bucket algorithm (GCRA) to pace the channel-sustain cell rate. The host must initialize PCR, PCR fraction, SCR, SCR fraction, and BT (burst tolerance). 10 Peak and minimum cell rate pacing (UBR+ traffic). The host must initialize PCR, PCR fraction, MCR, MCR fraction, and MDA. 11 Reserved.
	12	AVCF	Auto VC off. Determines APC behavior when the last buffer associated with this VC has been sent and no more buffers are in the VC's TxBD table, 0 The APC does not remove this VC from the schedule table and continues to schedule it to transmit. 1 The APC removes this VC from the schedule table. To continue transmission after the host adds buffers for transmission, a new ATM TRANSMIT command is needed, which can be issued only after the CP clears the VCON bit. (Bit 13)
	13	VCON	Virtual channel is on Should be set by the host before it issues an ATM TRANSMIT command. When the host sets TCT[STPS] (stop transmit), the CP deactivates this channel and clears VCON when the channel is next encountered in the APC scheduling table. The host can issue another ATM TRANSMIT command only after the CP clears VCON.
	14–15	INTQ	Points to one of four interrupt queues available.

Table 36-8. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	0–12	—	Reserved, should be cleared during initialization.
	13–15	AAL	AAL type 000 AAL0 —Reassembly with no adaptation layer 001 AAL1 —ATM adaptation layer 1 010 AAL5 —ATM adaptation layer 5 011 AAL3/4 —ATM adaptation layer 3/4 100 AAL2 —ATM adaptation layer 2 101 AAL1_CES —ATM adaptation layer 1 with circuit emulation service All others reserved.
0x04	—	TxDBPTR	Tx data buffer pointer. Holds the real address of the current position in the Tx buffer.
0x08	—	TBDCNT	Transmit BD count. Counts the remaining data to transmit in the current transmit buffer. Its initial value is loaded from the data length field of the TxBD when a new buffer is open; its value is subtracted for any transmitted cell associated with this channel.
0x0A	—	TBD_Offset	Transmit BD offset. Holds offset from TBD_BASE of the current BD. Initialize to 0.
0x0C	0–7	Rate Reminder	Rate remainder. Used by the APC to hold the rate remainder after adding the pace fraction to the additive channel rate. Initialize to 0.
	8–15	PCR Fraction	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot. If this is an ABR channel, this field is automatically updated by the CP.
0x0E	—	PCR	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract. Note that for an ABR channel, the CP automatically updates PCR to the ACR value.
0x10	—	—	Protocol-specific
0x16	—	APCLC	APC linked channel. Used by the CP. Initialize to 0 (null pointer).
0x18	—	ATMCH	ATM cell header. Holds the full (4-byte) ATM cell header of the current channel. The transmitter appends ATMCH to the cell payload during transmission.

Table 36-8. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1C	0–1	—	Reserved, should be cleared during initialization.
	2–7	PMT	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE+PMT × 32. Can be changed on-the-fly.
	8–15	TBD_BASE	TxBD base. Points to the first BD in the channel's TxBD table. The 8 most-significant bits of the address are taken from BD_BASE_EXT in the parameter RAM. The four least-significant bits of the address are taken as zero.
0x1E	0–11		
	12	BNM	Buffer-not-ready interrupt mask. Can be changed on-the-fly. 0 The transmit buffer-not-ready event of this channel is masked. (TBNR event is not sent to the interrupt queue.) 1 The buffer-not-ready event of this channel is enabled.
	13	STPT	Stop transmit. Initialize to 0. When the host sets this bit, the CP deactivates this channel and clears TCT[VCON] when the channel is next encountered in the APC scheduling table. Note that for AAL5 if STPT is set and frame transmission is already started (TCT[INF] = 1), an abort indication will be sent (last cell with zero length field).
	14	IMK	Interrupt mask. Can be changed on-the-fly. 0 The transmit buffer event of this channel is masked. (TXB event is not sent to the interrupt queue.) 1 The transmit buffer event of this channel is enabled.
	15	PM	Performance monitoring. Can be changed on-the-fly. 0 No performance monitoring for this VC. 1 Performance is monitored for this VC. When a cell is sent for this VC, the performance monitoring table indicated in PMT field is updated.

36.9.2.1 AAL1 Protocol-Specific TCT

Figure 36-23 shows the AAL1 protocol-specific TCT.

Local Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x10	—		Valid Octet Size (VOS)						PFM	SRT	SPIF	STF	—	SN		
0x12	SRTS_DEV								Block Size							
0x14	ICASB/SRTS_TMP								Structured Pointer (SP)							

Figure 36-23. AAL1 Protocol-Specific TCT

Table 36-9 describes AAL1 protocol-specific TCT fields.

Table 36-9. AAL1-Specific TCT Field Descriptions

Offset	Bits	Name	Description
0x10	0–1	—	Reserved, should be cleared during initialization.
	2–7	VOS	Valid octet size. Partially filled cell mode only. Specifies the number of valid octets from the beginning of the AAL1 user data field. For unstructured service, values 1-47 are valid; for structured service, values 1-46 are valid.
	8	PFM	Partially filled mode. 0 Partially filled cells mode is not used. 1 Partially filled cells mode is used. The transmitter copies only valid octets from the buffer to the AAL1 cell. The size of the valid octets from the beginning of the AAL1 user data field is specified in the VOS (valid octet size) field.
	9	SRT	Synchronous residual time stamp. Unstructured format only. The MPC8560 supports SRTS generation using external logic. If this mode is enabled, the MPC8560 reads the SRTS from external logic and inserts it into four cells for which SN = 1, 3, 5, or 7. The MPC8560 reads the new SRTS from external logic every eight cells. (See Section , “.”) 0 SRTS mode is not used. 1 SRTS mode is used.
	10	SPIF	Structured pointer inserted flag. Indicates that a structured pointer has been inserted in the current cycle. The user should initialize this field to zero. **Used by the CP only**.
	11	STF	Structured format. 0 Unstructured format is used. 1 Structured format is used. Note that although the structured format may be used, if the block size is one, the user should clear STF so that only non P-format AAL1 cells are generated.
	12	—	Reserved, should be cleared during initialization.
	13–15	SN	Sequence number field. Used by the CP to check the incoming cells SN. Initialize to 0.
0x12	0–3	SRTS_DEV	Used to select a SRTS device. The SRTS device address is SRTS_BASE[0–27]+SRTS_DEV[28:31]. SRTS_BASE is taken from the parameter RAM and is 16-byte aligned.
	4–15	Block Size	Used only in structured format. Specifies the structured block size (Block Size = 0xFFF = 4 Kbytes maximum). Note that when working in CAS mode (CESM = 1 and CASM = 1), the block size should be programmed to the superframe block size plus the size of the CAS block. However, when working in basic mode (CES without CAS: CESM = 1 and CASM = 0) the block size should be programmed to the number of MCC slots (Nx64) per frame assigned to this channel.
0x14	0–3	ICASB/SRTS_TMP	ICASB applies when in CAS mode. Incoming CAS Block. Points to one of the eight available internal CAS block. The starting address of the table is IN_CAS_BLOCK_BASE + ICASB × 32. See 36.4.7.1, “CAS Routing Table” for more details. Note that the RCT and TCT use the same CAS routing table (CRT). SRTS_TMP applies when not in CAS mode. Before a cell with SN = 1 is sent, the CP reads the SRTS code from external SRTS logic, writes it to SRTS_TMP, and then inserts SRTS_TMP into the next four cells with an odd SN.
	4–15	SP	Structured pointer. Used by the CP to calculate the structured pointer. Initialize to 0. Structured format only.

36.10 Outgoing CAS Status Register (OCASSR)

Figure 36-24 shows the outgoing CAS block status register (OCASSR).

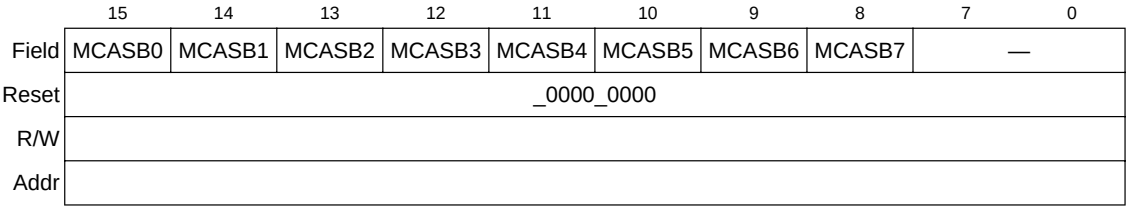


Figure 36-24. Outgoing CAS Status Register (OCASSR)

This status register contains flag bits that provide user software with an indication that the CP has modified the associated CAS block. When the ATM receiver operates in core CAS modify mode (RCT[CCASM] is set), the CP generates an interrupt and sets the appropriate bit in OCASSR each time an AAL1 cell is received with new signaling information (that is, one or more signaling nibble has changed).

Note that these flag bits are sticky, that is, they remain set until cleared by software. If new signaling is received and the relevant sticky bit is already set, the CP updates the CAS block without generating another interrupt.

Table 36-10 describes OCASSR fields.

Table 36-10. OCASSR Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared during initialization.
8, 9, 10, 11, 12, 13, 14, 15	MCASB*n*	Modify CAS Block *n*. When the CP updates this outgoing CAS block, it sets the MCASB*n* sticky bit and generates an interrupt to notify the core that the signaling information has changed in block *n*. Each channel selects the CAS block number in its RCT; see Section 36.9.1.1, “AAL1 Protocol-Specific RCT.”

36.11 Buffer Descriptors

AAL1 controller is a multi-channel protocol device which – for every channel – segments, reassembles, and tranceives data between several sets of memory buffers simultaneously. This complexity requires a separate list of BDs for each channel. Every channel is configured with two BD lists; one each for transmit and receive operations. The number of BDs in the table is user defined.

The BD table is a circular list, with the last entry specified by setting its table wrap bit. When the MPC8560 reaches the last BD, it returns to the beginning of the list. Each BD in the TxBD table points to a buffer to send. At the receive side, the user allocates dedicated buffers to each channel (that is, one BD for each buffer) . When the receiver or transmitter

completes writing or reading the buffer, it moves to the next buffer in the list and optionally issues an interrupt.

36.11.1 Transmit Buffer Operation

The user prepares a table of BDs pointing to the buffers to be sent. The address of the first BD is put in the channel's TCT[TBD_BASE]. The transmit process starts when the core issues an ATM transmit command. The CP reads the first TxBD in the table and sends its associated buffer. When the current buffer is finished, the CP increments TBD_OFFSET, which holds the offset from TBD_BASE to the current BD. It then reads the next BD in the table. If the BD is ready (TxBD[R] is set), the CP continues sending. If the current BD is not ready, the CP polls the ready bit at the channel rate unless TCT[AVCF] is set, in which case the CP removes the channel from the APC and clears TCT[VCON]. The core must issue a new ATM transmit command to restart transmission.

Note that when the ATM transmitter is in CES mode, the buffer-not-ready state is ignored by the ATM controller; see Section 36.4.1.2, "TDM-to-ATM."

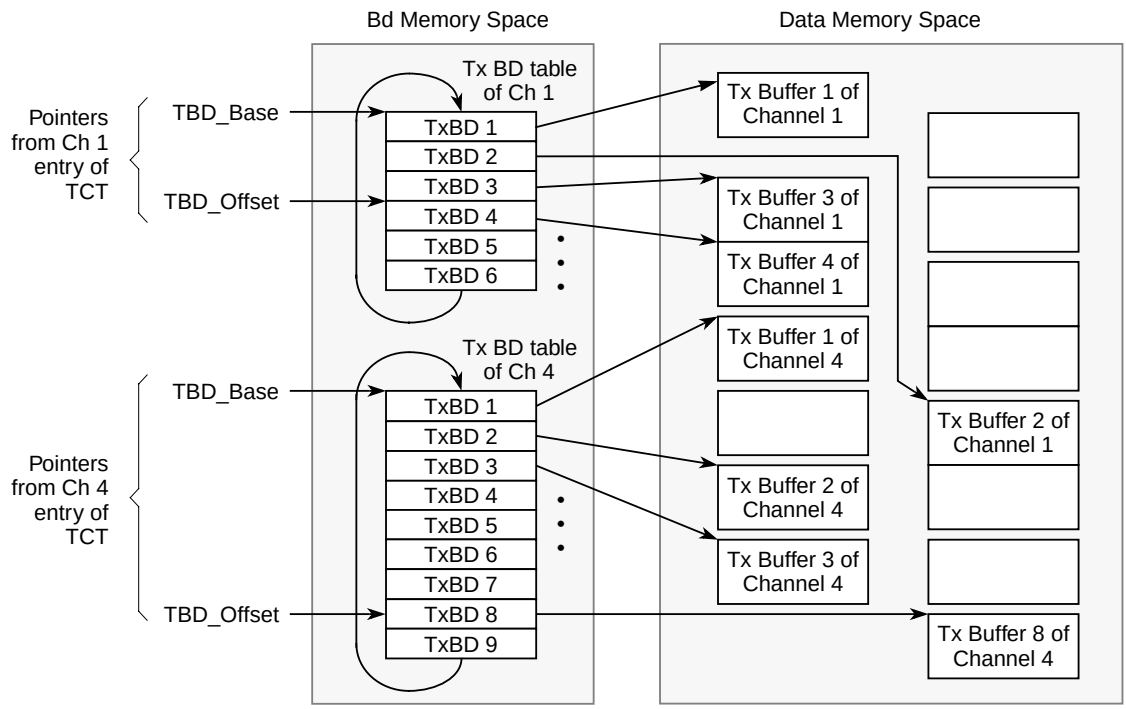


Figure 36-25. Transmit Buffers and BD Table Example

36.11.2 Receive Buffer Operation

The user prepares a table of BDs pointing to the receive buffers. The address of the first BD is put in the channel's RCT[RBD_BASE]. When an ATM cell arrives, the CP opens the first

BD in the table and starts filling its associated buffer with received data. When the current buffer is full, the CP increments RBD_OFFSET, which is the offset to the current BD from RBD_BASE, and reads the next BD in the table. If the BD is empty (RxBD[E] = 1), the CP continues receiving. If the BD is not empty, a busy condition has occurred and the ATM receiver optionally issues an interrupt to the event queue.

Note that when the ATM receiver is in CES mode, the buffer-not-ready (busy) state is handled by an automatic slip control mechanism; see Section 36.5, “ATM-to-TDM Adaptive Slip Control.”

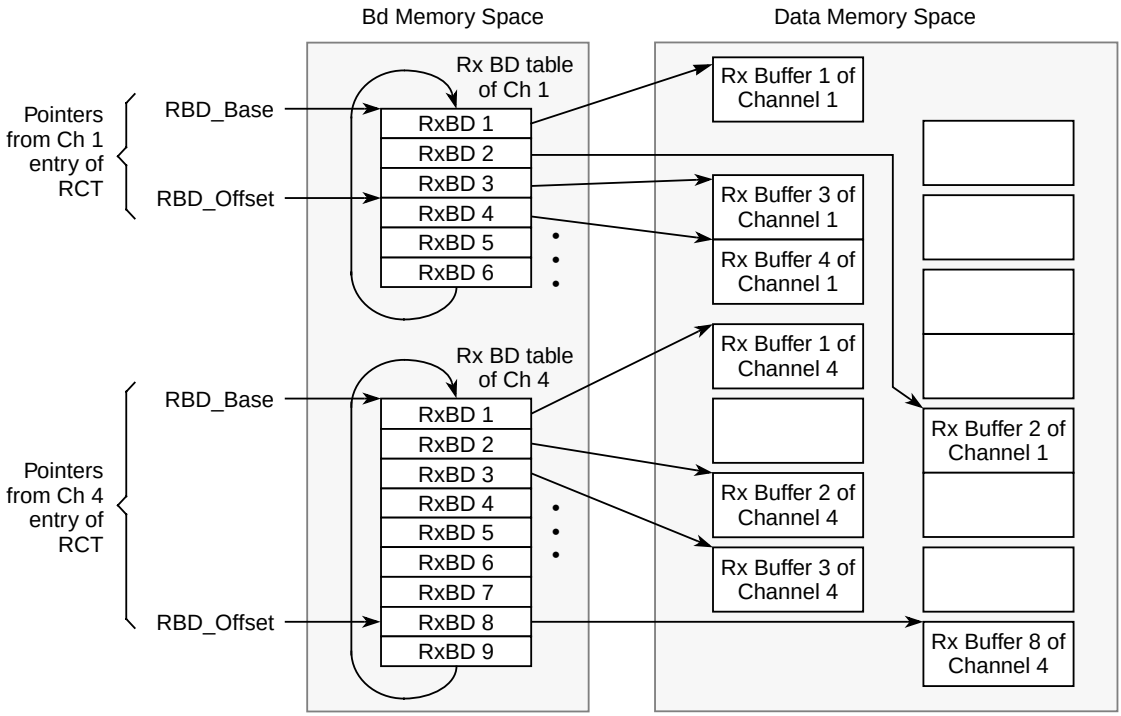


Figure 36-26. Receive Buffers and BD Table Example

36.12 ATM Controller Buffers

Table 36-11 describes properties of the ATM receive and transmit buffers.

Table 36-11. Receive and Transmit Buffers

AAL	Receive		Transmit	
	Size	Alignment	Size	Alignment
AAL5	Multiple of 48 octets (except last buffer in frame)	Double word aligned	Any	No requirement
AAL3/4	At least 44 octets (except last buffer in frame)	Double word aligned	At least 44 octets	No requirement

Table 36-11. Receive and Transmit Buffers

AAL	Receive		Transmit	
	Size	Alignment	Size	Alignment
AAL1	Multiple of 8 octets	No requirement	Multiple of 8 octets	No requirement
AAL0	52-64 octets.	Burst-aligned	52–64 octets.	No requirement

36.12.1AAL1 RxBD

Figure 36-27 shows the AAL1 RxBD.

Local Offset	0	1	2	3	4	5	6	7	8	9	10	11 15
0x00	E	—	W	I	—	—	CM	—	—	EOSF		—
0x02	Data Length											
0x06	Rx Data Buffer Pointer											
0x08												

Figure 36-27. AAL1 RxBD

Table 36-12 describes AAL1 RxBD fields.

Table 36-12. AAL1 RxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	E	Empty. 0 The buffer associated with this RxBD is filled with received data or data reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP cannot use this BD again while E = 0. 1 The buffer is not full. This RxBD and its associated receive buffer are owned by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	—	—
	2	W	Wrap (final BD in table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of this current channel. After this buffer is used, the CP receives incoming data into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table overall space is constrained to 64 Kbytes.
	3	I	Interrupt 0 No interrupt is generated after this buffer has been used. 1 An Rx buffer event is sent to the interrupt queue after the ATM controller uses this buffer. FCCE[GINT*x*] is set when the INT_CNT reaches the global interrupt threshold.
	4–5	—	—
	6	CM	• Continuous mode 0 Normal operation. 1 The empty bit (RxBD[E]) is not cleared by the CP after this BD is closed, allowing the associated buffer to be overwritten automatically when the CP next accesses this BD.
	7–8	—	—
	9	EOSF	End of super frame. CES mode (RCT[CESM = 1]) only. 0 No signaling information should be inserted after closing this buffer. 1 When closing this buffer, the ATM receiver unpacks the CAS information from the incoming AAL1 cells and stores it in the internal CAS block. Note that this bit should be set by the user at the end of each super-frame in the common (MCC, ATM) BD table. See Section 36.4.6, “Channel Associated Signaling (CAS) Support.”
	10–15	—	—
0x02	—	DL	Data length. The number of octets the CP writes into the buffer once its BD is closed.
0x04	—	RXDBPTR	Rx data buffer pointer. Points to the first location of the associated buffer; may reside in either internal or external memory. This pointer must be burst-aligned.

36.12.2 AAL1 TxBDs

Figure 36-28 shows the AAL1 TxBD.

Local Offset	0	1	2	3	4	5	6	7	8	9	10	15
0x00	R	—	W	I	—		CM	—		EOSF	—	
0x02	Data Length (DL)											
0x04	Tx Data Buffer Pointer (TXDBPTR)											
0x06												

Figure 36-28. AAL1 TxBD

Table 36-13 describes AAL1 TxBD fields.

Table 36-13. AAL1 TxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	R	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The CP clears this bit after the buffer has been sent or after an error condition is encountered. 1 The buffer prepared for transmission by the user has not been sent or is being sent. No fields of this BD may be written by the user once R is set.
	1	—	—
	2	W	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer is used, the CP sends outgoing data from the first BD in the table (the BD pointed to by the channel's TCT[TBD_BASE]). The number of TxBDs in this table is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	I	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx buffer event is sent to the interrupt queue after this buffer is serviced. FCCE[GINT*x*] is set when the INT_CNT counter reaches the global interrupt threshold.
	4–5	—	—
	6	CM	Continuous mode 0 Normal operation. 1 The CP does not clear the ready bit after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD.
	7–8	—	—
	9	EOSF	End of super frame. CES mode (TCT[CESM = 1]) only. 0 No signaling information should be inserted after closing this buffer. 1 When closing this buffer the ATM transmitter fetches the CAS information from the internal CAS block and packs it to the outgoing AAL1 cells. Note that this bit should be set by the user at the end of each super-frame in the common (MCC, ATM) BD table. See Section 36.4.6, "Channel Associated Signaling (CAS) Support."
	10–15	—	—

Table 36-13. AAL1 TxBD Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	—	DL	The number of octets the ATM controller should transmit from this BD's buffer. It is not modified by the CP. The value of DL should be greater than zero.
0x04	—	TXDBPTR	Tx data buffer pointer. Points to the address of the associated buffer. The buffer may reside in either internal or external memory. This value is not modified by the CP.

36.13 AAL1 Exceptions

There are four circular interrupt queues available for each channel. The interrupt queue number is programmed in RCT[INTQ] and TCT[INTQ]. Events can be masked by clearing interrupt mask bits in the RCT and TCT.

When one of the AAL1 channels generates an interrupt request, the CP writes a new entry to the table consisting of the channel's number and a description of the exception. The valid (V) bit is then set and INTQ_PTR is incremented. When INTQ_PTR reaches the entry in which W is set, it returns to the first entry of the queue. Each one-word entry provides detailed interrupt information to the host. More details can be found in Section 29.11, "ATM Exceptions."

36.13.1 AAL1 Interrupt Queue Entry

Figure 36-29 shows an interrupt queue entry.

Local Offset	0	1	2	3 – 7	8	9	10	11	12	13	14	15
0x00	V	—	W	—	SLIPE	SLIPS	CASUP	TBNR	RXF	BSY	TXB	RXB
0x02	Channel Code (CC)											

Figure 36-29. AAL1 Interrupt Queue Entry

Table 36-14 describes interrupt queue entry fields.

Table 36-14. AAL1 Interrupt Queue Entry Field Descriptions

Offset	Bits	Name	Description
0x00	0	V	Valid interrupt entry 0 This interrupt queue entry is free and can be use by the CP. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	—	—
	2	W	Wrap bit. When set, this is the last interrupt circular table entry. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–7	—	—

Table 36-14. AAL1 Interrupt Queue Entry Field Descriptions (continued)

Offset	Bits	Name	Description
	8	SLIPE	Slip End.Set when an AAL1 channel exits a slip state (the channel's adaptive counter reaches the ATM_Start threshold and the ATM channel regains its SYNC). At this point the receiver starts to receive the incoming cells. See Section Figure 36-15., "Pre-Overrun Sequence." Note that this interrupt has an associated channel code and can be masked by clearing RCT[SLIPIM] as described in Section 36.9.1, "Receive Connection Table (RCT)."
	9	SLIPS	Slip Start.Set when an AAL1 channel enters a slip state (the channel's adaptive counter reaches the ATM_Stop threshold or the ATM channel loses its SYNC). At this point the receiver drops incoming cells until the adaptive counter reaches the ATM_Start threshold and the channel is resynchronized. See Section Figure 36-15., "Pre-Overrun Sequence." Note that this interrupt has an associated channel code and can be masked by clearing RCT[SLIPIM] as described in Section 36.9.1, "Receive Connection Table (RCT)."
	10	CASUP	CAS Update Interrupt. Set when one of the eight outgoing CAS blocks is updated by the CP. New signaling information has been received within the received AAL1 cells. Note that this interrupt has an associated channel code and is available in CES mode and when RCT[CCASM] is set.
	11	TBNR	Tx buffer-not-ready. Set when a transmit buffer-not-ready interrupt is issued. This interrupt has an associated channel code and is issued when the CP tries to open a TxBD that is not ready (R = 0). This interrupt is sent only if TCT[BNM] is set. Note that for AAL5, this interrupt is sent only if frame transmission is started. In this case, an abort frame transmission is sent (last cell with length = 0), the channel is taken out of the APC, and TCT[VCON] is cleared.
	12	RXF	Rx frame. RXF is set when an Rx frame interrupt is issued. This interrupt is issued at the end of AAL5 PDU reception. This interrupt has an associated channel code and is issued only if RCT[RXFM] is set.
	13	BSY	Busy condition. The BD table or the free buffer pool associated with this channel is busy. Cells were discarded due to this condition. This interrupt has an associated channel code.
	14	TXB	Tx buffer. TXB is set when a transmit buffer interrupt is issued. This interrupt has an associated channel code and is enabled when both TxBD[I] and TCT[IMK] are set.
	15	RXB	Rx buffer. RXB is set when an Rx buffer interrupt is issued. This interrupt has an associated channel code and is enabled when both RxBD[I] and RCT[RXBM] are set.
0x02	—	CC	Channel code specifies the channel associated with this interrupt.

36.14 AAL1 Sequence Number (SN) Protection Table

The 32-byte sequence number protection table, pointed to by AAL1_SNPT_BASE in the ATM parameter RAM, resides in dual-port RAM and is used for AAL1 only. The table should be initialized according to Table 36-15.

Table 36-15. AAL1 Sequence Number (SN) Protection Table

Local Offset	0 15
0x00	0x0000
0x02	0x0007

Table 36-15. AAL1 Sequence Number (SN) Protection Table (continued)

0x04	0x000D
0x06	0x000A
0x08	0x000E
0x0A	0x0009
0x0C	0x0003
0x0E	0x0004
0x10	0x000B
0x12	0x000C
0x14	0x0006
0x16	0x0001
0x18	0x0005
0x1A	0x0002
0x1C	0x0008
0x1E	0x000F

36.15 Internal AAL1 Statistics Tables

An AAL1 statistics table, shown in Table 36-16, resides in the dual-port RAM and holds AAL1 statistics on a per-VC basis. AAL1_Int_STATT_BASE points to the base address of these tables. Each AAL1 channel has its own table with a starting address given by AAL1_Int_STATT_BASE + ATM_CHANNEL# × 8.

Table 36-16. AAL1 DPR Statistics Table

Offset	Name	Width	Description
0x00	Rx_AAL1_VALID	Hword	16-bit cyclic counter. Counts the total received AAL1 cells delivered to the receive buffers. This counter includes the tag cells (with SCE, SNE).
0x02	Rx_AAL1_BOV	Hword	16-bit cyclic counter. Counts the number of ATM buffer-pre overrun events i.e the ATM write pointer reaches the ATM_STOP threshold.See Section 36.5, “ATM-to-TDM Adaptive Slip Control.”
0x04	Tx_AAL1_VALID	Hword	16-bit cyclic counter. Counts the transmitted AAL1 cells.
0x06	Tx_AAL1_BUN	Hword	16-bit cyclic counter. Counts the number of ATM buffer underrun events. See Section 36.4.1.2, “TDM-to-ATM.”

36.16 External AAL1 Statistics Tables

An AAL1 statistics table, shown in Table 36-17, resides in the external memory and holds AAL1 statistics on a per-VC basis. AAL1_Ext_STATT_BASE points to the base address of these tables. Each AAL1 channel has its own table with a starting address given by AAL1_Ext_STATT_BASE + ATM_CHANNEL# × 16.

Table 36-17. AAL1 External Statistics Table

Offset	Name	Width	Description
0x00	Rx_AAL1_LOST	Hword	16-bit cyclic counter. Counts the number of AAL1 lost cells events. See Section Figure 36-15., “Pre-Overrun Sequence.”
0x02	Rx_AAL1_MISS	Hword	16-bit cyclic counter. Counts the number of AAL1 misinserted events. See Section Figure 36-15., “Pre-Overrun Sequence.”
0x04	Rx_AAL1_SCE	Hword	16-bit cyclic counter. Counts the number of AAL1 sequence errors i.e the expected SC is not match with the received one (ESC! = RSC). See Section Figure 36-15., “Pre-Overrun Sequence.”
0x06	Rx_SNP_Error	Hword	16-bit cyclic counter. Counts the number of ATM cell that received with SNP error. (AAL1 PDU Header Error)
0x08	Rx_AAL1_SPE	Hword	16-bit cyclic counter. Counts the number of structured pointer error events (that is, parity error, tag cell, or pointer mismatch). See Section 36.7, “Pointer Verification Mechanism.”
0x0A	Rx_ReSYNC	Hword	16-bit cyclic counter. Counts the number of AAL1 resynchronized events: pointer reframes, slip events, two consecutive cells with errors (SNE, SCE, Tag…), and two consecutive pointers with errors (parity error or pointer mismatch).
0x0C–0x0E	—	Word	Reserved, should be cleared during initialization.

Note that both the internal and the external statistics tables should be cleared by the user.

36.17 SRTS Generation and Clock Recovery Using External SRTS Logic

The MPC8560 supports SRTS generation using external logic. If SRTS generation is enabled (TCT[SRT] = 1), the MPC8560 reads SRTS[0–3] from the external SRTS logic and inserts it into four cells whose SN fields are 1, 3, 5, and 7, as shown in Figure 36-30.

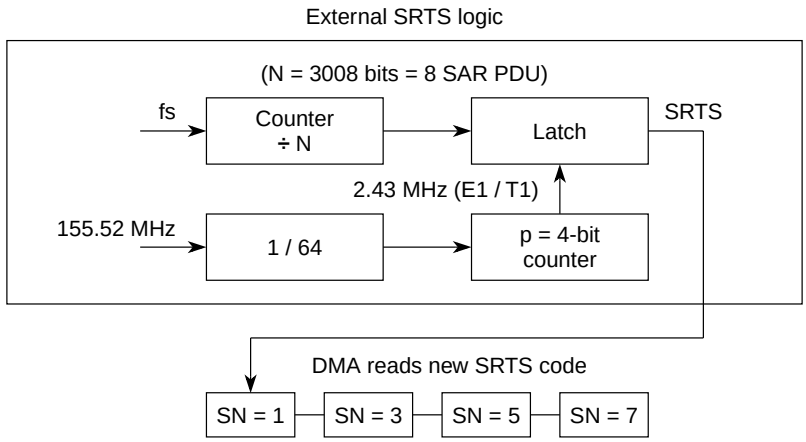


Figure 36-30. AAL1 SRTS Generation Using External Logic

For every eight cells, the external SRTS logic should supply a valid SRTS code. The CP reads the SRTS code using a DMA read cycle of 1-byte in size. Each AAL1 channel can be programmed to select one of 16 addresses available for reading the SRTS result; see TCT[SRTS_DEV] in Section 36.9.2.1, “AAL1 Protocol-Specific TCT.” The SRTS code should be placed on the least-significant nibble of that address (SRTS[0] = lsb, SRTS[3] = msb). The SRTS is synchronized with the sequence count cycle: SRTS[0] is inserted into the cell with SN = 7 and SRTS[3] is inserted into the cell with SN = 1. For every eighth AAL1 SAR PDU, the SRTS logic samples a new SRTS and stores it internally. The SRTS is a sample of a 4-bit counter with a 2.43-MHz reference clock (for E1/T1) synchronized with the network clock.

The MPC8560 supports clock recovery using an external SRTS PLL. If SRTS recovery is enabled (RCT[SRT] = 1), the MPC8560 tracks the SRTS from the four incoming cells whose SN fields are 1, 3, 5, and 7, and writes the result to external SRTS logic, as shown in Figure 36-31.

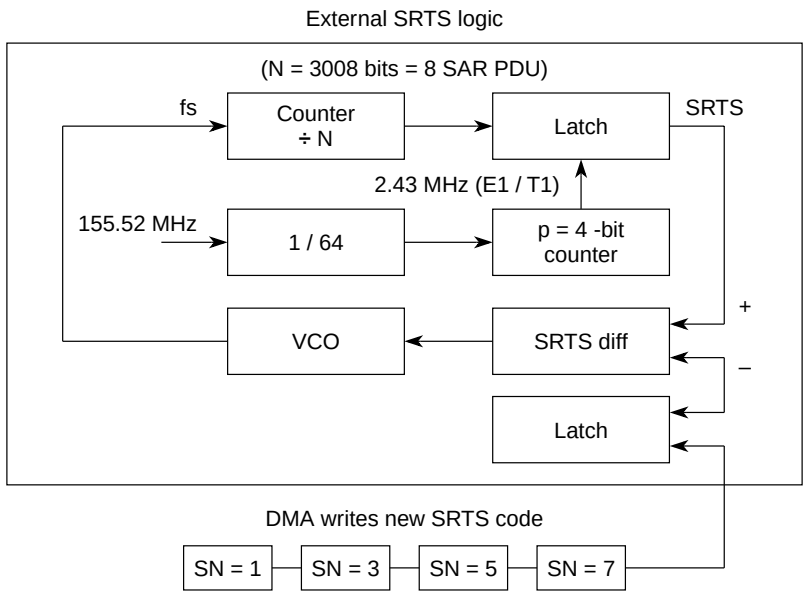


Figure 36-31. AAL1 SRTS Clock Recovery Using External Logic

On every eighth cell, the MPC8560 writes a new SRTS code to the external logic. The CP writes the SRTS code using a DMA write cycle of 1-byte in size. Each AAL1 channel can be programmed to select one of 16 addresses available for writing the SRTS result; see RCT[SRTS_DEV] in Section 36.9.1.1, “AAL1 Protocol-Specific RCT.” The SRTS code is written to the least-significant nibble of that address (SRTS[0] = lsb, SRTS[3] = msb). The SRTS is synchronized with the sequence count cycle: SRTS[3] is read from the cell with SN = 1 and SRTS[0] is read from the cell with SN = 7. The SRTS PLL makes periodic clock adjustments based on the difference between the locally generated SRTS and the remote SRTS collected every eight received cells.

36.18 CES-Specific Additions to the MCC

Additions to the MCC global parameter RAM and modifications to the channel-specific CHAMR and INTMASK registers have been implemented to support CES operation. This section describes the CES enhancements to the MCCs.

36.18.1 CES Additions to the MCC Parameter RAM

The global MCC parameters specific to CES operation are described in Table 36-19.

Table 36-18. CES-Specific Global MCC Parameters

Offset [1]	Name	Width	Description
0x00	CATB	Hword	CES adaptive threshold tables base address. Points to the dual-port RAM area containing the CES slip control thresholds and the adaptive counter See 36.5, “ATM-to-TDM Adaptive Slip Control.” Should be 8-byte aligned (8 octets for each AAL1-MCC channel). User-defined and should match the CATB value programmed in the AAL-1 parameter RAM; see Section 36.8.1, “AAL1 Parameter RAM.”
0x02	—	Hword	Reserved, should be cleared during initialization.
0x04, 0x08, 0x0C, 0x10	UTAa, UTAb, UTAc, UTAd,	Hword	Underrun template address for TDM*x*. Points to the dual-port RAM area containing the user-defined template to be sent during an MCC transmitter pre-underrun condition.
0x06, 0x0A, 0x0E, 0x12	UTSa, UTSb, UTSc, UTSd,	Hword	Underrun template size for TDM*x*. This is the size in bytes of the underrun template buffer.

[1] The offset to the CES-specific global MCC parameter RAM for MCC1 is 0x8780. For MCC2, it is 0x8880.

36.18.2 Channel Mode Register (CHAMR)—CES Mode

Figure 36-32 shows the user-initialized channel mode register, CHAMR, for CES operation. It is the same as the CHAMR in transparent mode with three extra CES fields in bits 13–15.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	MODE	POL	—		EP	RD	SYNC		—		TS	RQN		CESM	UDC	UTM
Reset	—															
R/W	R/W															
Offset	0x1A															

Figure 36-32. Channel Mode Register (CHAMR)—CES Mode

The CHAMR in CES mode fields are described in Table 36-19.

Table 36-19. CHAMR Field Descriptions—CES Mode

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>MODE</td><td>Channel mode. Selects either HDLC or transparent mode. Must be cleared for CES operation.
0 Transparent mode.
1 HDLC mode</td></tr>
<tr><td>1</td><td>POL</td><td>Enable polling. POL enables the transmitter to poll the TxBDs.
0 Polling is disabled (The CPM does not access the external bus to check the R bit in the TxBD).
1 Polling is enabled.
POL can be used to optimize the use of the external bus. Software should always set POL at the beginning of a transmit sequence of one or more frames. The CP clears POL when no more buffers are ready in the transmit queue, i.e. when it finds a BD with R = 0 (for example, at the end of a frame or at the end of a multi-frame transmission). To prevent a significant number of useless transactions on the external bus, software should always prepare the new BD, or multiple BDs, and set BD[R] before enabling polling.</td></tr>
<tr><td>2–3</td><td>—</td><td>Reserved. Must be set.</td></tr>
<tr><td>4</td><td>EP</td><td>Empty polarity and enable polling.
• 0The E bit in the RxBD is handled in positive logic (1 = empty; 0 = not empty). Polling occurs only if POL is set.
• 1The E bit in the RxBD is handled in negative logic (0 = empty, 1 = not empty). Polling occurs disregarding the value of POL.</td></tr>
<tr><td>5</td><td>RD</td><td>0 Normal bit order (transmit/receive the lsb of each octet first)
1 Reversed bit order to be reversed (transmit/receive the msb of each octet first).</td></tr>
<tr><td>6–7</td><td>SYNC</td><td>Synchronization. SYNC controls synchronization of multi-channel operation in transparent mode.
<table>
<tr><th>SYNC</th><th>Receive</th><th>Transmit</th><th>Description</th></tr>
<tr><td>00</td><td>None</td><td>None</td><td>Transmitter and receiver operate with no synchronization algorithm</td></tr>
<tr><td>01</td><td>Slot</td><td>Slot</td><td>The first data is sent/received in the slot defined in the slot assignment table (for super channels only)</td></tr>
<tr><td>10</td><td>8-bit</td><td>None</td><td>Receive data synchronization uses an 8-bit pattern specified by the 8 msb of RCVSYNC. The sync bytes will not be written to the receive buffer</td></tr>
<tr><td>11</td><td>16-bit</td><td>None</td><td>Receive data synchronization uses a 16-bit pattern specified by RCVSYNC. The first byte of the sync pattern will not be written to the receive buffer. The second byte of the sync pattern will be written to the receive buffer (first and second represent the order in which the two bytes of the sync pattern are received on the serial channel).</td></tr>
</table></td></tr>
<tr><td>8–9</td><td>—</td><td>Reserved, should be cleared during initialization.</td></tr>
<tr><td>10</td><td>TS</td><td>Receive time stamp. If this bit is set a 4 byte time stamp is written at the beginning of every data buffer that the BD points to.If this bit is set the data buffer must start from an address equal to 8*N-4 (N is any number larger than 0).</td></tr>
<tr><td>11–12</td><td>RQN</td><td>Receive queue number. Specifies the receive interrupt queue number.
• 00Queue number 0.
• 01Queue number 1.
• 10Queue number 2.
• 11Queue number 3.</td></tr>
<tr><td>13</td><td>CESM</td><td>Circuit emulation service mode.
• 0Normal mode
• 1CES mode</td></tr>
</table>

Table 36-19. CHAMR Field Descriptions—CES Mode (continued)

Bits	Name	Description
14	UDC	User-defined cell support. • 0User-defined ATM cells are not supported. • 1User-defined ATM cells are supported.
15	UTM	Underrun template mode. • 0Retransmit the last buffer. • 1Send the user-defined template.

36.18.3 Interrupt Table and Mask (INTMSK)

Two interrupts related to CES operation have been added to the MCC transmitter interrupt table entries. The corresponding mask bits are in the interrupt mask (INTMSK). Bits 4 and 5 (SLIPE and SLIPS) provide slip indications for the MCC transmitter. Figure 36-33 shows the CES interrupts and their mask bits.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Interrupt Entry	—				SLIPE	SLIPS	UN	TXB	—		NID	IDL	MRF	RXF	BSY	RXB
INTMSK	—				CES Mask Bits		Mask Bits		—		Mask Bits					

Figure 36-33. MCC Interrupt Table Entry with CES Slip Indicators and INTMSK

The two MCC transmitter slip indicators are described in Table 36-20.

Table 36-20. CES Slip Indicators in the MCC Interrupt Table Entries

Bits	Name	Description
4	SLIPE	Slip End. Set when an MCC channel interworking with an ATM channel exits the slip state (the connection's CESAC falls to the MCC_Start threshold). At this point, the transmitter stops sending the underrun template (or last buffer) and starts sending valid data.
5	SLIPS	Slip Start. Set when an MCC channel interworking with an ATM channel enters a slip state (the channel's CESAC reaches the MCC_Stop threshold). At this point the transmitter freezes and begins sending the underrun template (or last buffer) until CESAC falls to the MCC_Start threshold.

36.19 Application Considerations

- The buffer size (MCC [MRBLR]) must be a multiple of 8 octets.
- The CAS buffer operates in continuous mode with newer signaling information continuously overwriting older signaling.
- The CES application does not use super-frame synchronization for the data flow in ATM-to-TDM and TDM-to-ATM interworking. However, for the signaling flow, the MPC8560 uses the super-frame sync signal to know when to supply the signaling information to the external framer. The external framer then places the signaling information at the appropriate position in the super frame. See Section 36.4.6, "Channel Associated Signaling (CAS) Support."

- Simple external logic is needed to synchronize the MCC-to-framer serial connection. When going from TDM-to-ATM, the CAS information should be read by the MCC on the 24th frame of a super frame.
- The adaptive rate FIFO method can be implemented by the core by calculating the difference between the ATM and MCC pointers.
- When going from TDM-to-ATM, the ATM channel should be programmed to work at a higher rate than the MCC super-channel rate. We expect that the jitter caused by the APC traffic reshaping will depend on the ATM channel rate (PCR, PCR_FRACTION). Figure 36-34 illustrates this timing issue. See also Section 36.4.1.2, “TDM-to-ATM.”

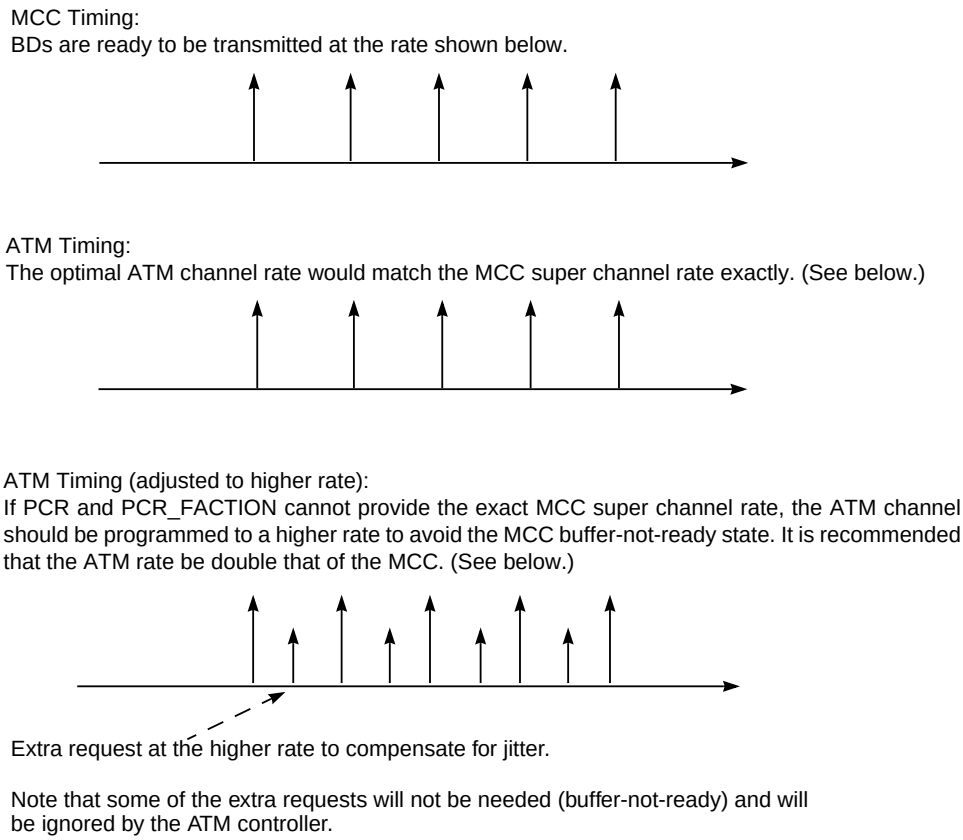


Figure 36-34. TDM-to-ATM Timing Issue

Chapter 37
AAL2

NOTE

This chapter includes some functionality not available strictly in the ROM of the MPC8560 silicon. CPS switching and VBR support is available only as part of a RAM microcode package.

The microcode implementation of the ATM adaptation layer type 2 (AAL2) on the MPC8560 is compliant with the ITU-T recommendations I.363.2 and I.366.1. This section describes the functionality and data structures of AAL2 CPS, CPS switching, and SSSAR, and is intended as a supplement to the Chapter 35, “ATM Controller.”

37.1 Introduction

AAL2 enables the multiplexing of voice and data channels over a single ATM VC. The channels consist of packets transported within individual ATM cells (see Figure 37-1). Packet lengths are allowed to vary in order to accommodate bandwidth fluctuations of the individual channels. Each packet has a channel identifier (CID) so that each AAL2 user (channel) is uniquely identified by the triplet VP | VC | CID.

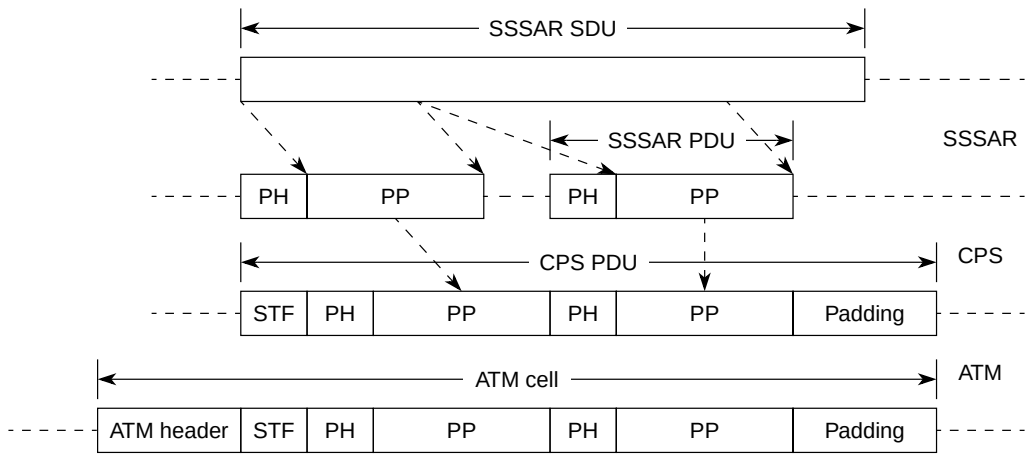


Figure 37-1. AAL2 Data Units

AAL2 is subdivided into two sublayers (see Figure 37-2): the common part sublayer (CPS) and the service-specific convergence sublayer (SSCS). In the CPS sublayer, variable length

packets coming from multiple users are assembled into CPS-PDUs belonging to a single ATM VCC. The SSCS sublayer handles the mapping of user data to the CPS sublayer. The SSCS segments large data frames into smaller CPS packets and also provides different services to the user, such as transmission error detection. The SSCS sublayer is further divided into three service-specific layers: the service-specific segmentation and reassembly sublayer (SSSAR), the service-specific transmission error detection sublayer (SSTED), and the service-specific assured data transfer sublayer (SSADT).

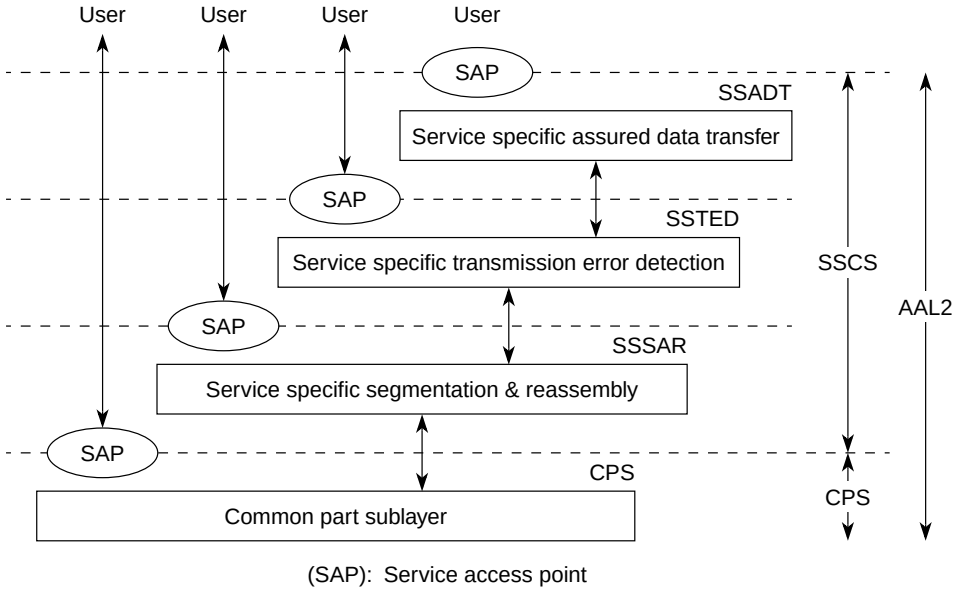


Figure 37-2. AAL2 Sublayer Structure

The AAL2 microcode implements the CPS and SSSAR sublayers. (The SSADT and SSTED sublayers are not implemented.) As shown in Figure 37-2, the user can access the CPS sublayer directly or through the SSSAR sublayer. The SSSAR sublayer is used mainly for transferring large data frames.

The AAL2 microcode also enables switching from one PHY | VP | VC | CID combination to another; an example is shown in Figure 37-3.

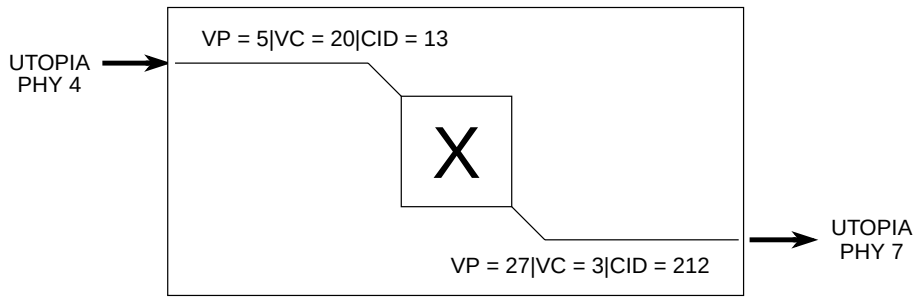


Figure 37-3. AAL2 Switching Example

37.2 Features

- Fully complies with ITU-T I.363.2 (09/97 and 11/00) and ITU-T I.366.1 (06/98) specifications.
- Supports up to 128 internal channels of AAL2
- Number of AAL2 external channels supported is subject to internal memory constraints
  - Each external channel requires space for at least on Transmit QUEUE Descriptor in internal memory. Typically, up to 1000 external channels can be supported.
- Supports CBR, VBR and UBR + traffic types.
  - PCR pacing (with optional Timer_CU)
  - VBR pacing (with optional Timer_CU)
  - UBR + pacing (no Timer_CU support)
- Priority mechanism for transmitting per VC. The priority mechanism provides for TX queues having equal or differing priorities. The SSSAR TX queues can be prioritized flexibly among the CPS TX queues.
- Timer_CU support.
- NoSTF mode support.
- Support for partially filled cells.
- User-defined cells (as described in the Section 36.7, “User-Defined Cells (UDC) of the Chapter 35, “ATM Controller,” in the *MPC8560 User’s Manual*).
- Interrupt indications include the ATM channel number, the CID, and the event type. The events reported are TX buffer not ready, TX buffer transmitted, RX buffer not ready, RX buffer, RX SSSAR frame, and RX AAL2 error events.
- CPS switching
  - Switching from a receive $PHY_1 | VP_1 | VC_1 | CID_1$ combination to another transmit $PHY_2 | VP_2 | VC_2 | CID_2$ combination.
  - Partial packet discard support.
  - For each switched queue, a counter for the total number of packets in the queue is available.
- CPS Receiver
  - Segmentation of CPS PDU directly to external memory queues.
  - A separate queue for every VP | VC | CID or a common queue for multiple VP | VC | CID combinations.
  - Receive one or multiple VP | VC | CIDs directly to a specific TX queue to enable switching.
  - Sequence number (SN) protection check for CPS-PDU.
  - CRC5 (HEC) check to detect errors in the CPS-PH of the CPS-Packet

- OSF (offset field) of the STF (start of frame) check (a valid value is less than 48)
- An SDU length limit parameter (Max_SDU_Deliver_Length) per ATM VC.
- Odd parity check for the STF octet of the CPS-PDU.

- CPS Transmitter
  - Reassemble CPS PDU directly from external memory.
  - Perform CPS-PDU padding as needed.
  - Insert Sequence Number bit of the CPS-PDU.
  - Parity bit is calculated to provide odd parity over the STF octet.
  - Calculation of CRC-5 on the first 19 bits of the CPS-PH.
  - UUI field in the CPS-Packet header is programmed according to the value of the CPS_UUI parameter (per packet).
  - A free running counter (per TX Queue) is decremented for each packet sent.
- SSSAR Receiver
  - Reassemble CPS packets from the same CID into an SSSAR SDU.
  - A separate queue for every PHY | VP | VC | CID.
  - Perform all the above mentioned CPS receiver functions.
  - A Ras_Timer mode is provided. When the Ras_Timer expires the buffer is closed with a timer expired error. The next packet received starts a new SSSAR SDU in a new buffer.
  - The SDU length is checked. If the frame exceeds the length limit (SSSAR_Max_SDU_Length), the receiver discards the rest of the packets from the current frame, closes the buffer and reports a Max_SDU violation error in the BD. The next packet received starts a new SSSAR SDU in a new buffer.
  - Partial Packet Discard. If no buffer is available when a packet arrives, the receiver enters a frame hunt state and discards each incoming packet from the current frame.
  - The UUI field is stored for the host into the RxBD after receiving the whole SSSAR frame.
- SSSAR Transmitter
  - Segmentation of SSSAR SDUs from SSSAR TX queue into CPS packets.
  - An SSSAR TX queue may contain several CIDs.
  - Performs all the above mentioned CPS transmitter functions.
  - A programmable segment length (Seg_Len) is copied from the SSSAR SDU into the CPS packet payload, except for when at the end of the frame or buffer.
  - UUI mode available. When UUI mode is enabled, the SSSAR UUI is copied from the byte following the last byte of the frame.

37.3 AAL2 Transmitter

The following sections describe the AAL2 transmitter.

37.3.1 Transmitter Overview

A transmitter cycle starts when the APC schedules an ATM channel number for transmission. The TCT is fetched and the AAL type of the channel is checked. For AAL2 cells, the transmitter first handles uncompleted packets from the previous cell of the current CID (partial and split cases) by filling the beginning of the cell with the remainder of the last packet.

Then, the transmitter performs the priority mechanism (see Section 37.3.2, “Transmit Priority Mechanism”) in order to fill the cell with new packets. The priority mechanism determines the order in which the TX queues are serviced. The transmitter continues to search for ready packets in the TX Queues until either the cell is successfully filled with packets, or no more packets are ready but the cell is not yet completed. In the first case the cell is simply sent. In the latter case, the optional Timer_CU (described in Section 37.3.5.1, “AAL2 Transmit Connection Tables”) is examined. If the Timer_CU has expired, the uncompleted cell is padded with zeros and sent; otherwise, the cell is temporarily stored in external memory for the CP to attempt to complete it the next time the channel is scheduled.

The TX queues are the data structures that store the CPS packets and SSSAR frames. Each TX queue can contain different CIDs. Each TX queue is maintained by a Tx queue descriptor (TxQD) that holds the queue pointer and parameters to manage the queue.

When the transmitter fetches a packet out of an SSSAR TX Queue, it usually takes out of the SSSAR buffer a number of octets equal to TxQD[Seg_Len] (see Section 37.3.5.4, “SSSAR Tx Queue Descriptor). The channel CID is taken from the BD of the first buffer of the SSSAR frame (see Section 37.3.5.5, “SSSAR Transmit Buffer Descriptor”). A CPS UUI = 27 is used for all the in-frame packets until the last packet from the SSSAR frame is sent. The last packet can optionally contain a per frame, user-defined UUI. After an SSSAR buffer is completely sent, an optional interrupt event is issued to the host. Also, if an SSSAR TX queue is empty an optional interrupt event is issued to the host.

In case of CPS TX Queue, the transmitter fetches the packet header out of a Buffer Descriptor and the packet payload out of a CPS buffer (see Section 37.3.5.3, “CPS Buffer Structure”). The HEC in the packet header is calculated by the CP. After a CPS packet is sent, an optional interrupt event is issued to the host. Also, if the CPS TX queue is empty an optional interrupt event is issued to the host.

The optional partial filled mode (see Section 37.3.3, “Partial Fill Mode (PFM)”) limits the number of data octets per cell. This can be used to ensure that a cell does not contain a split packet or to limit transmission to one packet per TX cell by setting a low partial fill threshold (PFT).

The no-STF (no start of frame) mode (see Section 37.3.4, “No STF Mode”) enables the transmission of cells that do not include the STF byte, thus allowing for 48-byte packets.

37.3.2 Transmit Priority Mechanism

The transmit priority mechanism operates in two modes:

- Round robin (TCT[Fix] = 0).
- Fixed priority (TCT[Fix] = 1).

The following sections describe the priority options.

37.3.2.1 Round Robin Priority

In round robin priority mode, the Tx queues all have equal priority. The transmitter starts with the TxQD pointed to by TCT[FirstQueue], as shown in Figure 37-4. The number of packets that the transmitter services from each queue is determined by the one-packet bit (TCT[OneP]). If TCT[OneP] = 0, the transmitter tries to process as many packets in the queue as needed to fill up the cell. Only when the queue is empty does the transmitter move on to the next queue (assuming the cell is not completed). If TCT[OneP] = 1, the transmitter attempts to take only one packet out of each queue. (Set TCT[OneP] for implementations where each queue contains only one CID.)

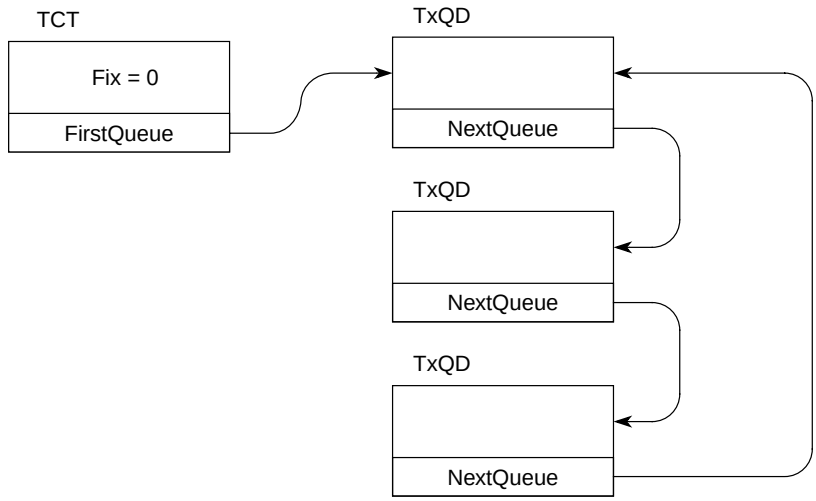


Figure 37-4. Round Robin Priority

The transmitter steps from one TxQD to the next along the queue links. The TCT[MaxStep] parameter limits the number of TX Queues that the transmitter visits during a cell time. If MaxStep is reached before the cell has been completely filled, one of the following events takes place:

- TCT[ET] = 0 (Timer CU disabled). The cell is padded with zeros and sent.

- TCT[ET] = 1 (Timer CU enabled). If the timer has not expired, the cell is not sent. (The transmitter attempts to fill the cell the next time this channel is scheduled.) If the timer has expired, the cell is padded with zeros and sent

After the transmitter sends a cell, it saves the queue link of the last TxQD serviced in TCT[FirstQueue].

37.3.2.2 Fixed Priority

In fixed priority mode (TCT[Fix] = 1), the transmitter, with each new cell, starts with searching the highest priority queue and then moves on to the lower priority queues. The TCT[FirstQueue] points to the highest priority queue, as shown in Figure 37-5, and remains unchanged by the CP.

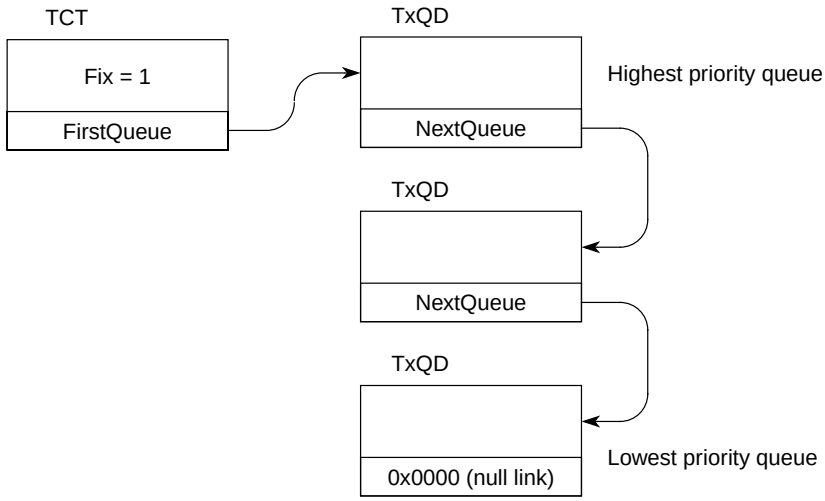


Figure 37-5. Fixed Priority Mode

The TCT[OneP] determines the number of packets that the transmitter attempts to take from each queue (see the explanation in round robin mode).

The NextQueue field of the lowest priority TxQD should be cleared (null link), and TCT[MaxStep] should be programmed to the total number of TX queues in the channel. When the transmitter reaches the null link and the cell is still not complete, the transmitter checks the TIMERCU mode as described above for the round robin mode.

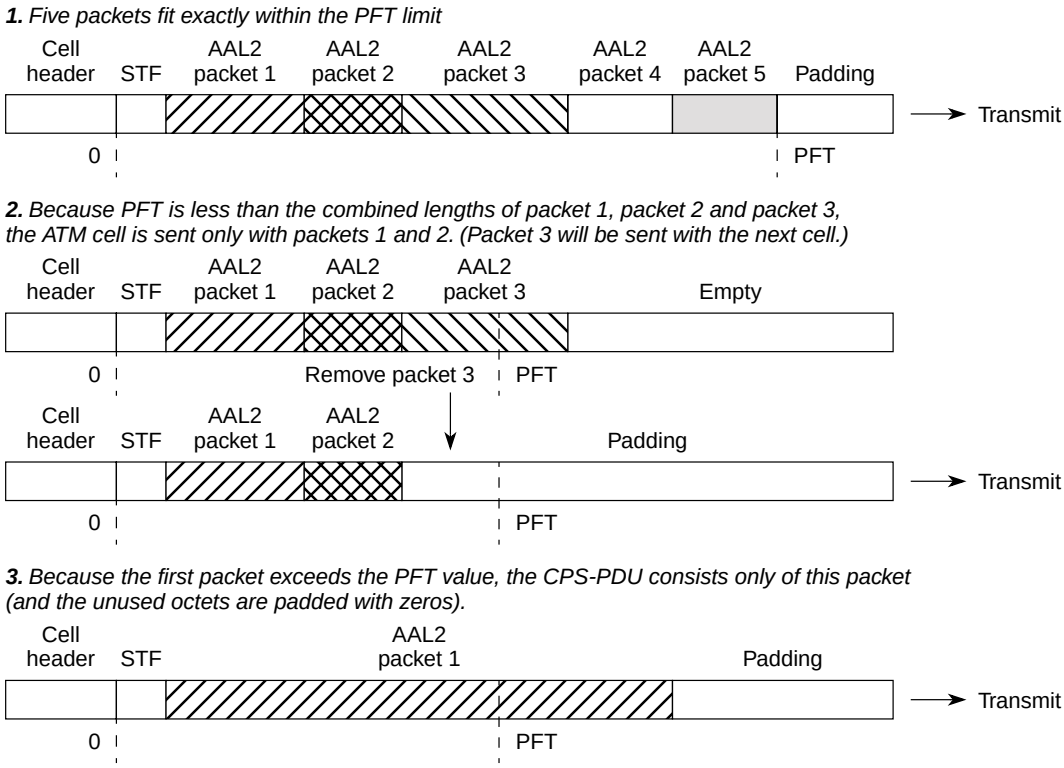
37.3.3 Partial Fill Mode (PFM)

The partial fill mode (TCT[PFM] = 1) allows the user to specify a partial fill threshold (TCT[PFT]), which limits the number of data octets sent with each ATM cell. Partial fill mode assures that packets are not split over two cells, unless the first packet of the cell is greater than 47 bytes.

In partial fill mode, the transmitter starts by filling the ATM cell with the first packet. After the first packet, the CP determines if including the next packet in the cell would exceed the PFT limit. If so, the second packet is not inserted into the current cell, the unused payload is padded with zeros, and the cell is sent. If the second packet does not exceed PFT, the CP inserts it into the CPS_PDU and moves on to the third packet, and so on.

By programming PFT = 1, the partial fill mode can also serve to assure that only one packet is sent per cell.

The following figures provide examples of partial fill mode operation:



In example (1), five packets fit exactly within the PFT limit.

In example (2), because PFT is less than the combined lengths of packet1, packet2 and packet3, the ATM cell is sent only with packets 1 and 2. (Packet3 will be sent with the next cell.)

In example (3), because the first packet exceeds the PFT value, the CPS-PDU consists only of this packet (and the unused octets are padded with zeros).

37.3.4 No STF Mode

The no-STF (no start of frame) mode enables the transmission of 48-byte packets by not including the STF byte in the CPS PDU. The no-STF mode should always be used with PFT programmed to 47 in order to prevent split and partial packets. For the AAL2 channels that use this mode, packets must not be larger than a maximum packet size of 48.

The user activates this mode per ATM channel by:

- Setting TCT[NoSTF] = 1 and TCT[PFM] = 1,
- and establishing a partial fill threshold by programming TCT[PFT] = 47.

Figure 37-6 shows a cell using no-STF mode.

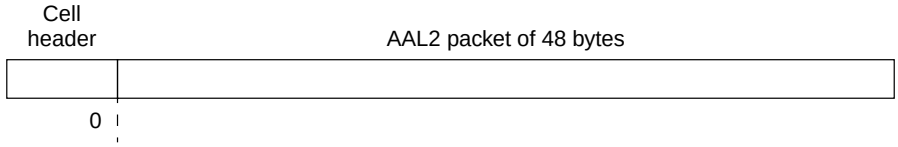


Figure 37-6. No Start of Frame (No-STF) Mode

37.3.5 AAL2 Tx Data Structures

The following sections describe the TCT and the structures in which CPS packets and SSSAR SDUs are stored in memory.

37.3.5.1 AAL2 Transmit Connection Tables

The transmit connection table (TCT) is a VC-level table and is where the AAL type for the ATM channel number is selected. The parameters related to the ATM channel number or to all the TX Queues of the ATM channel are maintained here. Figure 37-7 shows the AAL2-specific TCT.

	0 1	2	3 4	5	6	7	8	9	10 11	12	13	14 15
Offset + 0x00	—	GBL	BO	—	DTB	BIB	AVCF	PFM	ATT	ET	VCON	INTQ
Offset + 0x02	—									NoSTF	AAL	
Offset + 0x04	—											
Offset + 0x06												
Offset + 0x08	—											
Offset + 0x0A	FirstQueue											
Offset + 0x0C	Rate Remainder							PCR Fraction				
Offset + 0x0E	PCR											
Offset + 0x10	Timer_CU_period											
Offset + 0x12	Timer_Period_Shadow											
Offset + 0x14	—											
Offset + 0x16	APC Linked Channel											
Offset + 0x18	ATM Cell Header (VPI,VCI,PTI,CLP)											
Offset + 0x1a												
Offset + 0x1C	—	PMT					MaxStep					
Offset + 0x1E	—	PFT					—			OneP	STPT	Fix PM

Figure 37-7. AAL2 TCT

NOTE

When the channel is active, the CP fetches the TCT (32 bytes) using burst cycle and writes back only the first (24 bytes). Table 37-1 describes the AAL2 TCT fields.

Table 37-1. AAL2 TCT Field Descriptions

Offset	Bits	Name	Description
0x00	1	—	Reserved, should be cleared during initialization.
	2	GBL	Global. Setting GBL enables snooping of data buffers, BD, interrupt queues and free buffer pool.
	3–4	BO	Byte ordering. This field is used for data buffers. 0x Reserved. 1x Big endian.
	5	—	Reserved, should be cleared during initialization.
	6	DTB	Data buffer bus selection. 0 Data buffers reside on the system bus. 1 Data buffers reside on the local bus.
	7	BIB	Bus selection for the BDs, interrupt queues and the free buffer pool. 0 Reside on the system bus. 1 Reside on the local bus.
	8	AVCF	Auto VC off. Determines APC behavior when the last buffer associated with this VC has been sent and no more buffers are in the VC's TxBD table, 0 The APC does not remove this VC from the schedule table and continues to schedule it to transmit. 1 The APC removes this VC from the schedule table. To continue transmission after the host adds buffers for transmission, a new ATM TRANSMIT command is needed, which can be issued only after the CP clears the VCON bit. (Bit 13)
	9	PFM	Partial fill mode. See Section 37.3.3, "Partial Fill Mode (PFM)." 0 Partially filled cells are not supported. 1 Partially filled cells are supported.
	10–11	ATT	ATM traffic type 00 Peak cell-rate pacing (regular traffic). The host must initialize PCR and PCR fraction. Other traffic parameters are not used. 01 Peak and sustain cell rate pacing (VBR traffic). The APC performs a continuous-state leaky bucket algorithm (GCRA) to pace the channel-sustain cell rate. The host must initialize PCR, PCR fraction, SCR, SCR fraction, and BT (burst tolerance). 10 Peak and Minimum cell rate pacing. The host must initialize PCR, PCR fraction, MCR, MCR fraction, and MDA. 11 Reserved.
	12	ET	Enable Timer_CU. 0 Timer_CU operation in this channel is disabled. 1 Timer_CU operation in this channel is enabled.
	13	VCON	Virtual channel is on Should be set by the host before it issues an ATM TRANSMIT command. When the host sets the STPT (stop transmit) bit, the CP deactivates this channel and clears VCON. The host can issue another ATM TRANSMIT command only when the CP clears VCON.
	14–15	INTQ	Points to one of the four interrupt queues available.

Table 37-1. AAL2 TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	0–11	—	Reserved, should be cleared during initialization.
	12	NoSTF	No STF byte. See Section 37.3.4, “No STF Mode.” 0 Normal AAL2 cell structure. 1 The cell does not include the STF byte. In this mode each cell starts with a new packet and contains only whole packets (no split or partial).
	13–15	AAL	AAL type 000 AAL0 —Segmentation with no adaptation layer 001 AAL1 —ATM adaptation layer 1 protocol. (Use this configuration for rev A parts only, unless needed for backwards compatibility to existing software development.) 010 AAL5 —ATM adaptation layer 5 protocol 011 AAL3/4 —ATM adaptation layer 3/4 protocol 100 AAL2 —ATM adaptation layer 2 protocol 101 AAL1 with CES —ATM adaptation layer 1 protocol with optional support for circuit emulation service. (For rev B and later parts, this configuration is strongly recommended for all AAL1 applications, with or without CES.) All others reserved.
0x04		—	Reserved, should be cleared during initialization.
0x08		—	Reserved, should be cleared during initialization.
0x0A		FirstQueue	Points to the first queue to be serviced in the transmitter cycle. In round-robin priority mode (TCT[Fix] = 0), this pointer could point to any one of the TxQDs related to this channel. In fixed priority mode (TCT[Fix] = 1), this pointer should point to the highest priority TxQD.
0x0C	0–7	Rate Remainder	Rate remainder. Used by the APC to hold the rate remainder after adding the pace fraction to the additive channel rate. Should be cleared during initialization by the user.
	8–15	PCR Fraction	Peak cell rate fraction. Holds the peak cell rate fraction of this channel in units of 1/256 slot. If this is an ABR channel, this field is automatically updated by the CP.
0x0E	—	PCR	Peak cell rate. Holds the peak cell rate (in units of APC slots) permitted for this channel according to the traffic contract. Note that for an ABR channel, the CP automatically updates PCR to the ACR value.
0x10	—	Timer_CU_period	Timer_CU duration in units of APC slots. Assures that CPS-Packet already packed in a cell wait at most the Timer_CU duration. The user defines this field.
0x12	—	Timer_Period_Shadow	This field must be initialized to the Timer_CU period in units of APC slots.
0x14	0–15	—	Reserved, should be cleared during initialization.
0x16	—	APCLC	APC linked channel. Used by the CP. Should be cleared during initialization.
0x18	—	ATMCH	ATM cell header. Holds the full (4-byte) ATM cell header of the current channel. The transmitter appends ATMCH to the cell payload during transmission.

Table 37-1. AAL2 TCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x1C	0–1	—	Reserved, should be cleared during initialization.
	2–7	PMT	Performance monitoring table. Points to one of the available 64 performance monitoring tables. The starting address of the table is PMT_BASE + PMT x 32.
	8–15	MaxStep	Holds the number of TX Queues visited for each cell being prepared for transmission. In fixed priority mode (TCT[Fix] = 1), MaxStep should be set to the total number of TX queues in the channel. See Section 37.3.2, “Transmit Priority Mechanism.”
0x1E	0–1	—	Reserved, should be cleared during initialization.
	2–7	PFT	Partial fill threshold. Used for partially filled cells only; see Section 37.3.3, “Partial Fill Mode (PFM).” Specifies the maximum number of packet bytes allowed in a CPS PDU. The range 1–48 are valid values. If PFT = 48 in partial fill mode, performance is adversely affected. When not in partial fill mode, PFT must be initialized to 47.
	8–11	—	Reserved, should be cleared during initialization.
	12	OneP	One packet per queue. See Section 37.3.2, “Transmit Priority Mechanism.” 0 The transmitter reads as many packets as possible from each TX queue before moving to the next queue. 1 The transmitter reads only one packet from each TX queue before advancing to the next queue.
	13	STPT	Stop transmit. Should be cleared during initialization. When the host sets this bit, the CP removes this channel from the APC and clears TCT[VCON] flag.
	14	Fix	Fixed priority. See Section 37.3.2, “Transmit Priority Mechanism.” 0 Round robin priority. The TX Queues related to this channel all share the same priority. 1 Fixed priority. The TX Queues are ordered in a fixed priority ladder.
	15	PM	Performance monitoring 0 No performance monitoring for this VC. 1 Performance is monitored for this VC. When a cell is sent for this VC, the performance monitoring table indicated in PMT field is updated.

NOTE: All fields listed in this table must be initialized by the user.

37.3.5.2 CPS Tx Queue Descriptor

Each CPS TxBD table is managed by a CPS Tx queue descriptor (TxQD), as shown in Figure 37-8. The TxQD contains the address of the next BD to be serviced, and other queue-specific parameters. The NextQueue pointer is used to create a linked list of TxQDs, as described in Section 37.3.2, “Transmit Priority Mechanism.” The CPS TxQD is located in the dual-port RAM, in a 16-byte aligned address.

	0–7	8	9	10	11	12	13–15
Offset + 0x00	—	BNM	SW	HEC	CPS	TBM	—
Offset + 0x02	TxBD Table Offset In (switched mode only)						
Offset + 0x04	TxBD Table Base						
Offset + 0x06							
Offset + 0x08	TxBD Table Offset Out						
Offset + 0x0A	Number of Packets In Queue						
Offset + 0x0C	NextQueue						
Offset + 0x0E	—						

Figure 37-8. CPS Tx Queue Descriptor (TxQD)

Table 37-2 describes the CPS TxQD fields.

.

Table 37-2. CPS TxQD Field Descriptions

Offset	Bits	Name	Description
0x00	0–7	—	Reserved for internal use. (Used to save the BD status of the open BD.)
	8	BNM	Buffer not-ready interrupt mask of the TxBD table. 0 The transmit buffer-not-ready event for this queue is masked. (The event is not sent to the interrupt queue.) 1 The buffer-not-ready event for this queue is enabled.
	9	SW	Switching queue. 0 Normal TX Queue. 1 This TxQD handles a switching queue. The receiver and transmitter share this queue.
	10	HEC	HEC calculation. 0 Transmitter calculates the CPS header HEC. 1 The CPS header HEC is taken as is from the CPS buffer.
	11	CPS	Sublayer type. For a CPS TxQD, this field must be set. 0 SSSAR or SSTED. 1 CPS packet.
	12	TBM	Transmit buffer interrupt mask. 0 The transmit buffer event of this queue is masked. (The event is not sent to the interrupt queue). 1 The transmit buffer event of this queue is enabled.
	13–15	—	Reserved, should be cleared during initialization.
0x02	—	TxBD Table Offset In	Used only when this queue is used for switching (SW = 1). Should be cleared during initialization.
0x04	—	TxBD Table Base	This pointer points to the base address of the BD table.
0x08	—	TxBD Table Offset Out	Holds the offset from the TxBD table base to the next BD to be opened by the transmitter. Should be cleared during initialization.

Table 37-2. CPS TxQD Field Descriptions

Offset	Bits	Name	Description
0x0a	—	Number of Packets In Queue	Counts the number of packets currently in the queue. If this queue is switched, the receiver increments this counter with each new received packet and the transmitter decrements it with each packet sent. For switching, the user should initialize this counter to zero. When this queue is not switched, this counter counts down with every packet sent. (This can have various purposes such as evaluating the packet rate that is transmitted from this queue.).
0x0c	—	NextQueue	Points to the next TxQD to be serviced after this one. See Section 37.3.2, “Transmit Priority Mechanism.”
0x0e	—	—	Reserved, should be cleared during initialization.
NOTE: All fields listed in this table must be initialized by the user.			

37.3.5.3 CPS Buffer Structure

The CPS buffer structure consists of a BD table that points to data buffers. The BDs contain, apart from the buffer pointer, also the packet header. The buffers themselves contain the packet payload. See Figure 37-9.

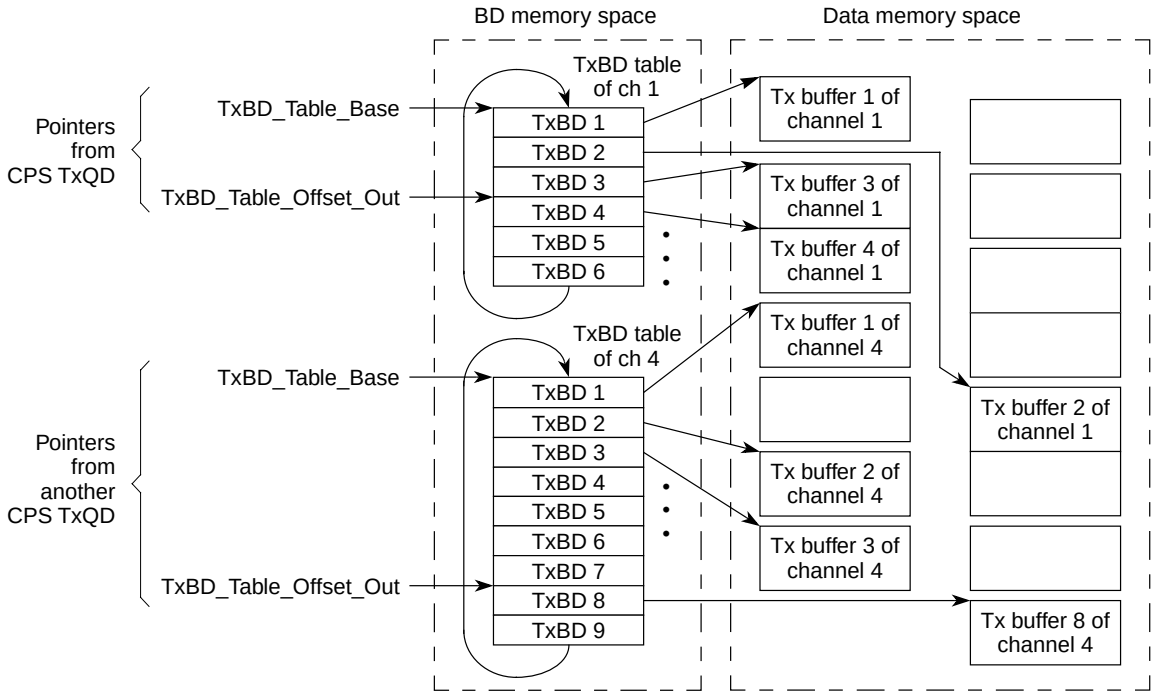


Figure 37-9. Buffer Structure Example for CPS Packets

Figure 37-10 shows a CPS TxBD.

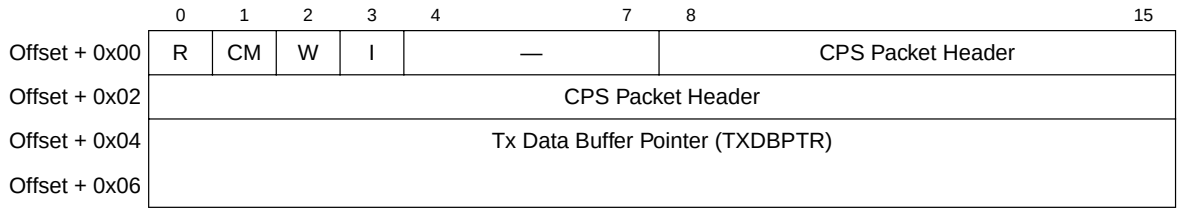


Figure 37-10. CPS TxBD

Table 37-3 describes the CPS TxBD fields.

Table 37-3. CPS TxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	R	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The CP clears R after the buffer is sent or after an error condition is encountered. 1 The user-prepared buffer has not been sent or is currently being sent. No fields of this BD may be written by the user once R is set.
	1	CM [1]	Continuous mode 0 Normal operation. 1 The CP does not clear R after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of the CM bit setting.
	2	W	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer is used, the CP transmits outgoing data for this channel from the first BD in the table (the BD pointed to by the channel's TxBD_table_Base in the TxQD). The number of TxBDs in this table is determined only by the W bit.
	3	I	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx Buffer event is sent to the interrupt queue after this buffer is serviced. The GHIN/GLIN bit in the event register is set when the INT_CNT counter reaches terminal count.
	4–7	—	Reserved, should be cleared during initialization.
	8–15	CPS Packet Header	This field contains the beginning (MSB) of the 3-byte packet header. See Figure 37-10 for the CPS packet header format.
0x02		CPS Packet Header	This field contains the rest of the packet header. If TxQD[HEC] = 0, the HEC part of the packet header is calculated by the CP, and the user may disregard the five least-significant bits of this field. See Figure 37-10 for the CPS packet header format.
0x04	—	TXDBPTR	Tx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This pointer is not modified by the CP.

[1] Setting continuous mode (TxBD[CM] = 1) is not allowed in CID switching mode.

All fields listed in this table must be initialized by the user.

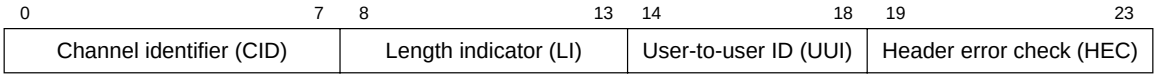


Figure 37-11. CPS Packet Header Format

37.3.5.4 SSSAR Tx Queue Descriptor

A SSSAR TxBD table and its associated buffers are collectively called an SSSAR TX Queue. Each SSSAR TX Queue is managed by an SSSAR TxQD, as shown in Figure 37-12. The TxQD contains the base address of the BD table, the offset of the next BD to be serviced, the data buffer pointer, and other queue-specific parameters. The NextQueue pointer is used to create a linked list of TxQDs, as described in Section 37.3.2, “Transmit Priority Mechanism.” The SSSAR TxQD is located in the dual-port RAM in a 32-byte aligned address.

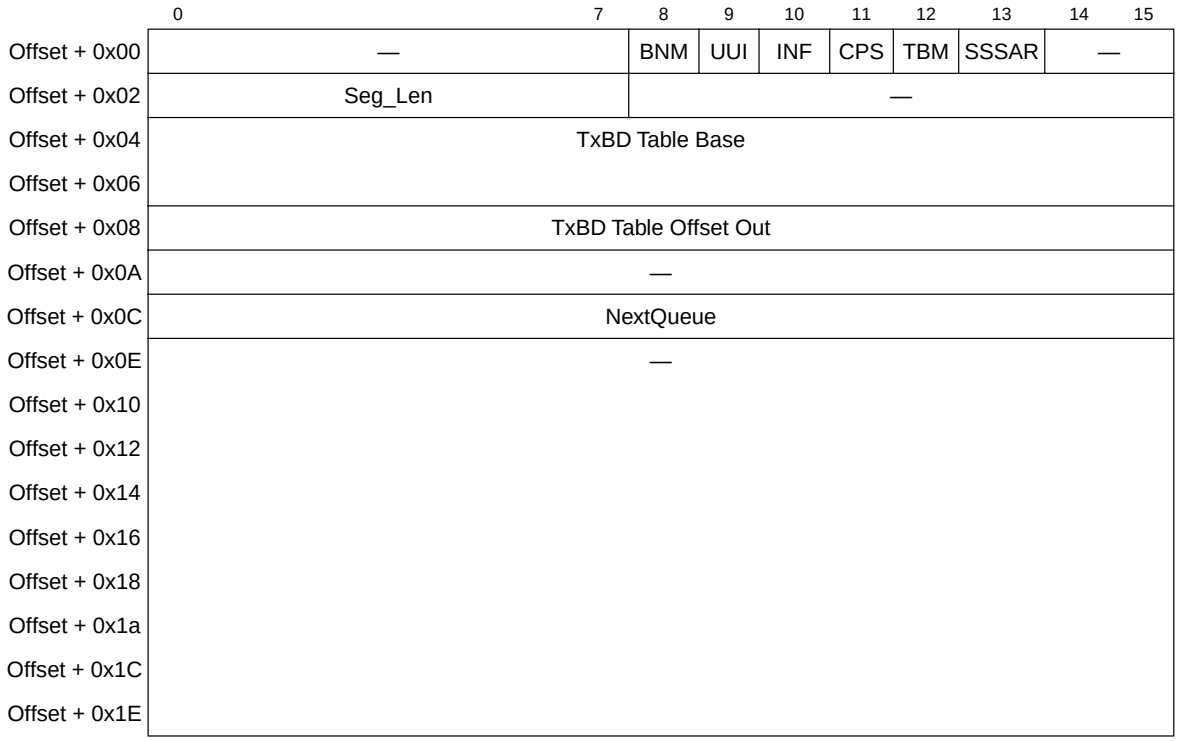


Figure 37-12. SSSAR Tx Queue Descriptor

Table 37-4 describes the SSSAR TxQD fields.

.

Table 37-4. SSSAR TxQD Field Descriptions

Offset	Bits	Name [1]	Description
0x00	0–7	—	Reserved, should be cleared during initialization.
	8	BNM	Buffer-not-ready interrupt mask of the TxBD table. 0 The transmit buffer-not-ready event for this queue is masked. (The event is not sent to the interrupt queue.) 1 The buffer-not-ready event for this queue is enabled.
	9	UUI	UUI insertion mode 0 UUI of last CPS packet is 0. 1 UUI of last CPS packet is taken from the next byte after the end of the buffer. [2]
	10	INF	Indicates the current state of the frame. 0 The next packet will be the first of a new frame. 1 Currently in the middle of the frame.
	11	CPS	Sublayer type. For an SSSAR TxQD, this field must be cleared. 0 SSSAR or SSTED. 1 CPS packet.
	12	TBM	Transmit buffer interrupt mask for TxBD table. 0 The transmit buffer event of this queue is masked. (The event is not sent to the interrupt queue.) 1 The transmit buffer event of this queue is enabled.
	13	SSSAR	SSSAR bit 0 SSTED sublayer 1 SSSAR sublayer
	14–15	—	Reserved, should be cleared during initialization.
0x02	0–7	Seg_Len	Specifies the maximum length in bytes of the SSSAR PDU (excluding the packet header). Seg_Len is limited to 45 in NoSTF = 1 mode. The CP always attempts to segment the SSSAR SDU according to this length, but not more than it.
	8–15	—	Reserved, should be cleared during initialization.
0x04	—	TxBD Table Base	Must be initialized to the first TxBD by the user.
0x08	—	TxBD Table Offset Out	Used to calculate the pointer to the next TxBD to be used for transmission. Should be cleared during initialization.
0x0C	—	NextQueue	Points to the next TxQD to be serviced after this one. See Section 37.3.2.
0x0E	—	—	Reserved, should be cleared during initialization.
0x10	—	—	
0x12	—	—	
0x14	—	—	
0x18	—	—	
0x1A	—	—	
0x1C	—	—	

[1] All fields listed in this table must be initialized by the user..

2 The 5-bit UUI field is inserted into the header of the last packet of an SSSAR SDU. The user must append the UUI to the last buffer as an additional byte. This additional byte is then inserted into the UUI field of the last packet header (note that, it is important that this additional byte—the byte after the last byte in the last data buffer of the SSSAR frame—contains the 5 bits of the UUI). The 3 MSB bits of this extra byte should be cleared; refer to Figure 37-10.

37.3.5.5 SSSAR Transmit Buffer Descriptor

The SSSAR buffer structure consists of a BD table that points to data buffers. The buffers may contain SSSAR SDUs belonging to different CIDs. Each buffer may contain a whole SSSAR SDU or part of it. The CPS CID is located in the first BD of the SSSAR SDU. See Figure 37-13.

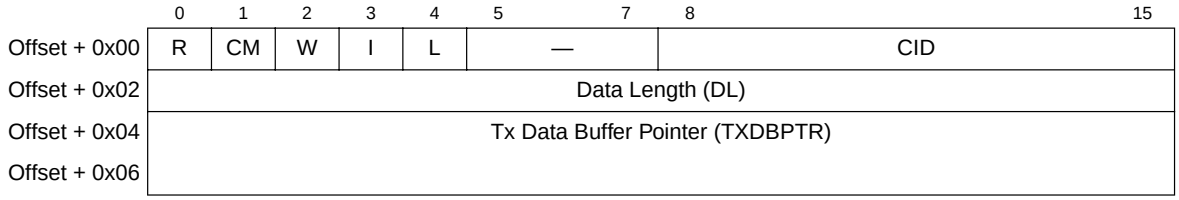


Figure 37-13. SSSAR TxBD

Table 37-5. SSSAR TxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	R	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated buffer. The CP clears R after the buffer is sent or after an error condition is encountered. 1 The user-prepared buffer has not been sent or is currently being sent. No fields of this BD may be written by the user once R is set.
	1	CM	Continuous mode 0 Normal operation. 1 The CP does not clear R after this BD is closed, allowing the associated buffer to be retransmitted automatically when the CP next accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of the CM bit setting.
	2	W	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer is used, the CP transmits outgoing data for this channel from the first BD in the table (the BD pointed to by the channel's TxBD_table_Base in the TxQD). The number of TxBDs in this table is determined only by the W bit.
	3	I	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 A Tx Buffer event is sent to the interrupt queue after this buffer is serviced. The GHIN/GLIN bit in the event register is set when the INT_CNT counter reaches terminal count.
	4	L	Last. 0 This is not the last buffer of the SSSAR SDU. 1 This is the last buffer of the SSSAR SDU.
	5–7	—	Reserved, should be cleared during initialization.
	8–15	CID	Contains the CID number of the SSSAR SDU pointed by this BD. This field should be written to the first BD of an SSSAR SDU.

Table 37-5. SSSAR TxBD Field Descriptions (continued)

Offset	Bits	Name	Description
0x02		**Data Length**	Contains the length of the buffer associated with this BD. If this is the last buffer (L = 1) and the UUI bit in the SSSAR TxQD is set, the 5-bit UUI field is located at (TXDBPTR + Data Length)[3:7] with bit [3] being the msb, that is, in the byte (right justified) immediately following the last byte of the buffer. For best bandwidth utilization and optimized partitioning of the SDU to packets of exactly Seglen size when an SDU is spread over multiple BD's, the application should set Data Length to be an integer multiple of Seg_Len. (Data Length == n x Seg_Len). For an SDU on a single BD this restriction does NOT apply.
0x04	—	**TXDBPTR**	Tx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the CP.
NOTE: Entries in boldface must be initialized by the user.			

37.4 AAL2 Receiver

The following sections describe the AAL2 receiver.

37.4.1 Receiver Overview

The receiver cycle starts after the FCC receives a cell. If the cell header is successfully mapped to an ATM channel number, the corresponding RCT is fetched and the AAL type is read. For AAL2 cells, the receiver begins by checking if the last cell received in this channel (CID) has an uncompleted (split) packet. If so, the receiver first finishes handling this packet. The UUI of the last packet of the SDU is stored in the RxBD where the L bit is set.

The receiver then goes through a validation process. The receiver matches the OSF field in the STF with the expected OSF based on the actual split packet (if the first packet is not split, the OSF should be zero). If the two values do not match, an OSF error interrupt is issued and the receiver drops the last packet. Also, if the STF parity check, the SN check or the OSF>47 check results in an error, the receiver issues an interrupt and discards the whole cell. If any of the above errors has occurred and the cell has started with the remainder of an uncompleted packet, the receiver does the following:

- For a CPS sublayer CID, the packet's RxBD[UP] (uncompleted packet) bit is set.
- For an SSSAR sublayer CID, the buffer is closed with RxBD[RxError = US = 10] (uncompleted SDU), and the rest of the frame is dropped.

The receiver now begins the process of extracting new CPS packets out of the cell with another round of error checking. The receiver examines each CPS packet header for the following errors:

- Incorrect packet HEC. The packet and rest of the cell are discarded.

- Packet length (LI + 1) is larger than CPS_Max_SDU_Length. The receiver discards the packet and then continues to extract the next packet in the cell. However, if the packet belongs to an SSSAR CID, the receiver closes the SSSAR buffer with RxBD[RxError = OS = 11] (oversized) and discards the rest of the frame.

The receiver issues an interrupt for each of the above errors.When a SSSAR buffer is closed with RxBD[RxError = US or OS], indicating Uncompleted SDU or Oversized, then RxBD[L] is set, and if RxQD[RFM] = 1 then the receiver also issues an RXF interrupt.

Then, if no errors have occurred in the packet header, the packet CID is used to match the PHY | VP | VC | CID with an RxQD; see Section 37.4.2, “Mapping of PHY | VP | VC | CID.” The match process yields an RxQD pointer. The RxQD indicates the type of SAR operation to be performed on the PHY | VP | VC | CID. Three SAR operation modes are supported:

- For the CPS sublayer, each packet from the CID is stored, as is, in a one packet buffer.
- For switching, the packet is stored directly into the transmit buffer, and the CID is translated.
- For the SSSAR sublayer, all packets received from the CID are reassembled into an SSSAR SDU similar to the BD and buffer structures used for AAL5 frames.
- For the SSSAR sublayer, last packet UUI indication is stored in the last RxBD of the SDU.

The RxQD also handles the buffers and BDs (in case of SSSAR) of the CID.

For the SSSAR sublayer, two additional parameters are verified for each new packet received:

- RAS_Timer expiration. The RAS_Timer_Duration defined in the AAL2 parameter RAM limits the time (starting when the first packet is received) allowed to receive a complete SSSAR SDU. If this time limit is exceeded, the receiver closes the current buffer with RxBD[RxError = TE = 01] (Ras_Timer expired) and starts a new SSSAR frame with the next packet. When a buffer is closed with RxBD[RxError = TE = 01], RxBD[L] is not set and the receiver does not issue an RXF interrupt.
- SSSAR_Max_SDU_Length. With each new packet the receiver checks whether the current accumulated length of the SSSAR SDU exceeds the SSSAR_Max_SDU_Length. If so, the receiver closes the current buffer with RxBD[RxError = OS = 11] (oversized), discards the rest of the SSSAR SDU, RxBD[L] is set, and if RxQD[RFM] = 1 issues an RXF interrupt.

NOTE

CIDs with the same number but from different AAL2 connections cannot use the same RxQD, unless they never have split packets.

37.4.2 Mapping of PHY | VP | VC | CID

The AAL2 mapping mechanism translates a PHY | VP | VC | CID combination into an RxQD. An RxQD can be unique per PHY | VP | VC | CID or can be common to several PHY | VP | VC | CIDs.

The mapping mechanism, shown in Figure 37-14, can be broken down as follows:

- Each ATM channel number (RCT) has its own CID mapping table. The mapping table can be placed in internal or external memory (according to RCT[MAP]) and is pointed to by RCT[CID Mapping Table Base]. The CID of the received packet is used as an index into the mapping table.
- Each entry in the mapping table contains a 2-byte RxQD offset. This offset, multiplied by 4, is the offset to an RxQD in either the internal or external RxQD table.
- The two RxQD tables serve all the ATM channel numbers of an FCC. (RxQD_Base_Int and RxQD_Base_Ext are defined in the FCC parameter RAM.)
- RxQD offsets from 8 through 511 point into the internal RxQD table located in dual-port RAM at RxQD_Base_Int. Note that the first 32 bytes of the internal RxQD table are reserved (so offsets 0–7 are reserved).
- RxQD offsets greater than 511 point into the external RxQD table located at RxQD_Base_Ext + (512 x 4).
- Because the three types of RxQDs are different sizes, some offset numbers may not be used.

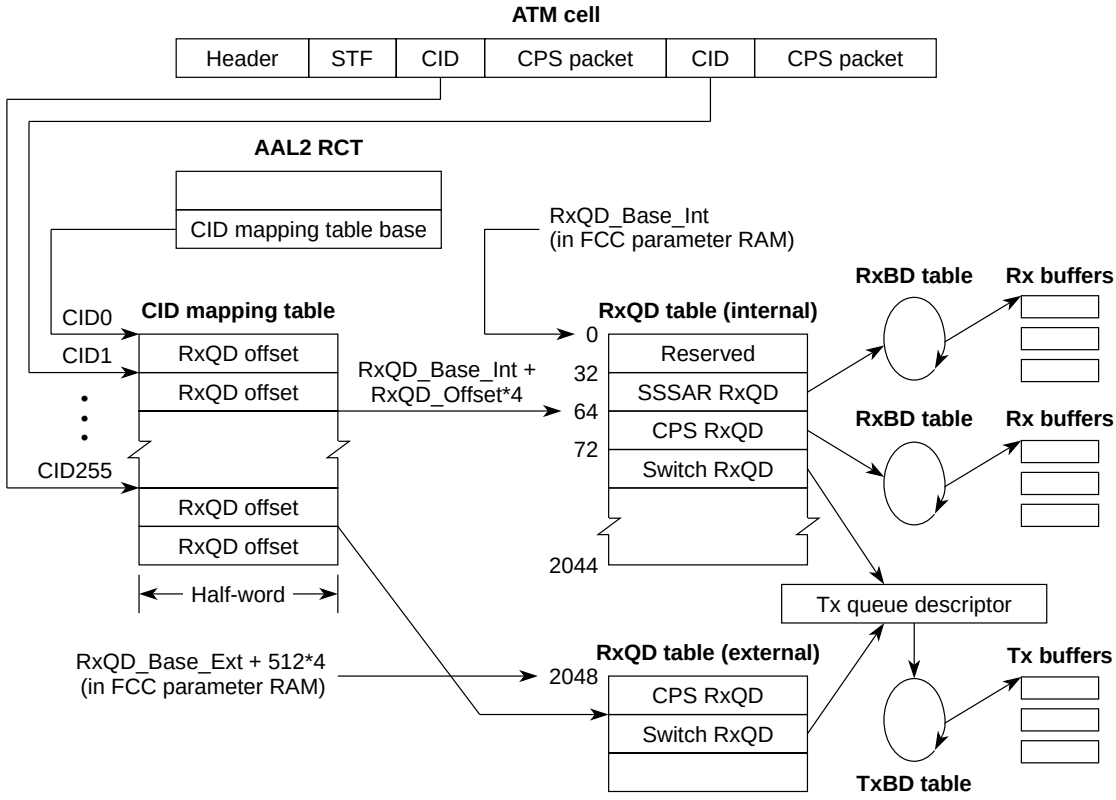


Figure 37-14. CID Mapping Process

37.4.3 AAL2 Switching

Switching is performed by pointing an RX CID at a switch RxQD (see Figure 37-15). The switch RxQD is unique for each Rx CID. The descriptor holds a translation CID number and a pointer to a CPS TxQD into which this packet is saved and later sent by the transmitter. (The TxQD pointer is responsible for the actual PHY | VP | VC switching.) The TxQD pointed to by the switch RxQD(s) should have TxQD[SW] set and should not be modified by the host when the channel is active. The transmit scheduling of the packet is done by the APC according to the programmed bit rate of the ATM channel that holds the switched queue.

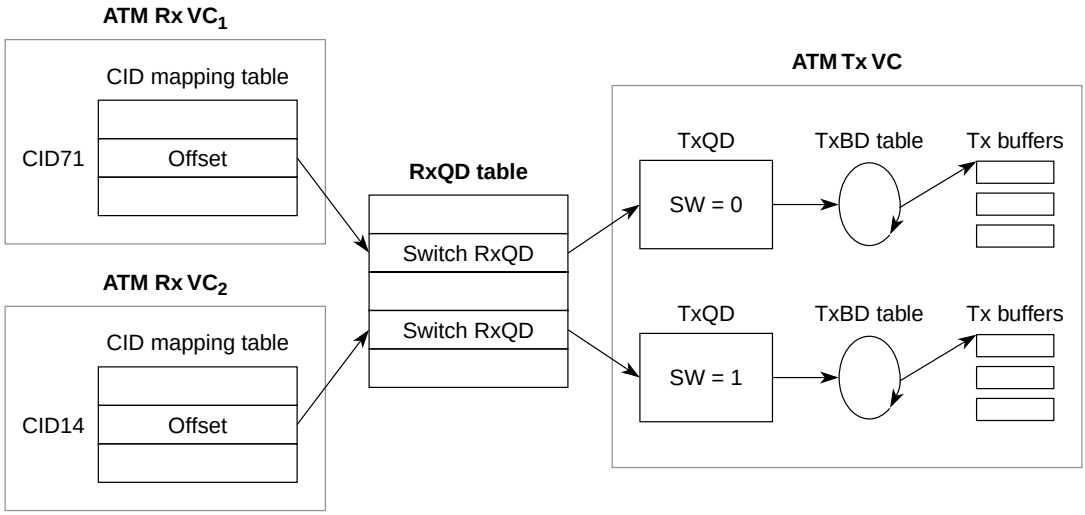


Figure 37-15. AAL2 Switching

A partial packet discard mode is provided for the AAL2 switched channels that perform end-to-end SSSAR. When this mode is enabled (switch RxQD[PPD] = 1), if no buffer is available to receive a packet in the middle of a frame, the subsequent middle packets of the SSSAR SDU are discarded. When the last packet of the SSSAR SDU arrives, the receiver attempts to re-open a buffer.

A number-of-packets-in-queue counter is available in the TxQD. The CP increments the counter for each packet received and decrements it for each packet sent. The host can poll the counter periodically to verify that the switching queues are not over-loaded.

37.4.4 AAL2 RX Data Structures

The following sections describe the RCT and the structures in which CPS packets and SSSAR SDUs are stored in memory.

37.4.4.1 AAL2 Receive Connection Tables

The receive connection table (RCT) is a VC-level table and is where the AAL type for the ATM channel number is selected. The parameters related to the ATM channel number or to all the RX Queues of the ATM channel are maintained here. The RCT also contains the pointer to the CID mapping table for the ATM VC. Figure 37-15 shows the AAL2-specific RCT.

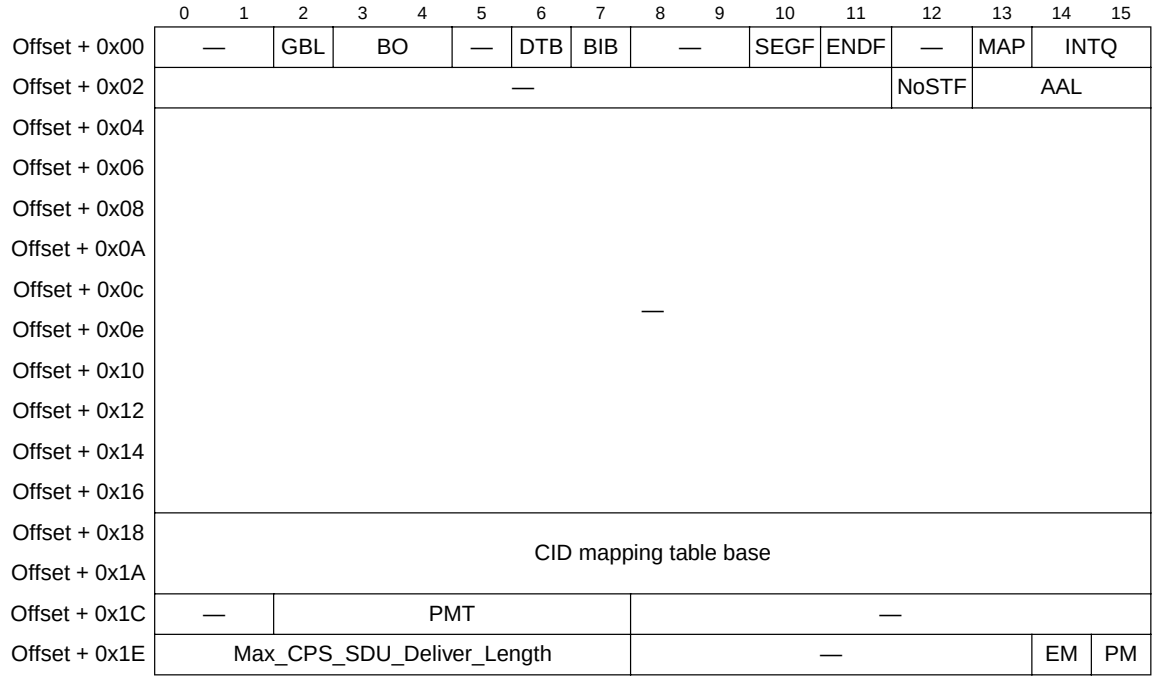


Figure 37-16. AAL2 Receive Connection Table (RCT)

NOTE

For an active channel, the CP uses a burst cycle to fetch the 32-byte RCT and writes back only the first 24 bytes. Table 37-6 describes the AAL2 RCT fields.

Table 37-6. AAL2 RCT Field Descriptions

Offset	Bits	Name	Description
0x00	0–1	—	Reserved, should be cleared during initialization.
	2	GBL	Global. Setting GBL enables snooping of data buffers, BDs, interrupt queues and free buffer pool.
	3–4	BO	Byte ordering—used for data buffers. 0x Reserved 1x Big endian
	5	—	Reserved, should be cleared during initialization.
	6	DTB	Data buffer bus selection. 0 Reside on the system bus. 1 Reside on the local bus.
	7	BIB	Bus selection for the BDs, interrupt queues, CID mapping table, RxQDs, and the free buffer pool. 0 Reside on the system bus. 1 Reside on the local bus.
	8–9	—	Reserved, should be cleared during initialization.
	10	SEGF	OAM F5 segment filtering 0 Do not send cells with PTI = 100 to the raw cell queue. 1 Send cells with PTI = 100 to the raw cell queue.
	11	ENDF	OAM F5 end-to-end filtering 0 Do not send cells with PTI = 101 to the raw cell queue. 1 Send cells with PTI = 101 to the raw cell queue.
	13	MAP	CID mapping table memory location select 0 Resides in the dual-port RAM. 1 Resides in external memory.
	14–15	INTQ	Assigns one of the four available interrupt queues to this ATM channel number.
0x02	0–11	—	Reserved, should be cleared during initialization.
	12	NoSTF	No STF byte. 0 Normal AAL2 cell structure. 1 The cell does not include the STF byte. In this mode each cell starts with a packet and contains only whole packets (no split or part).
	13–15	AAL	AAL type 000 AAL0 —Reassembly with no adaptation layer 001 AAL1 —ATM adaptation layer 1 protocol. (Use this configuration for rev A parts only, unless needed for backwards compatibility to existing software development.) 010 AAL5 —ATM adaptation layer 5 protocol 011 AAL3/4 —ATM adaptation layer 3/4 protocol 100 AAL2 —ATM adaptation layer 2 protocol 101 AAL1 with CES —ATM adaptation layer 1 protocol with optional support for circuit emulation service. (For rev B and later parts, this configuration is strongly recommended for all AAL1 applications, with or without CES.) All others reserved.
0x04	—	—	Reserved, should be cleared during initialization.
0x08	—		
0x0C	—		

Table 37-6. AAL2 RCT Field Descriptions (continued)

Offset	Bits	Name	Description
0x18	—	CID Mapping Table Base	Points to the base address of the CID mapping table (see Figure 37-14). If RCT[MAP] = 0, the pointer contains a dual-port RAM address and only the 16 lsb (at 0x1A and 0x1B) are relevant. If MAP = 1, the pointer is a full 32-bit address in the memory space.
0x1C	0–1	—	Reserved, should be cleared during initialization.
	2–7	PMT	Performance monitoring table. Assigns one of the available 64 performance monitoring tables to this VC. The table's starting address is PMT_BASE + PMT x 32.
	8–15	—	Reserved, should be cleared during initialization.
0x1E	0–7	Max_CPS_SDU_Deliver_Length	Indicates the maximum size CPS_SDU in bytes that is allowed to be transported on this channel. This value is compared to the length of each CPS_SDU before it is delivered, as specified in the ITU-T recommendation I.363.2.
	8–13	—	Reserved, should be cleared during initialization.
	14	EM	Receive error mask for AAL2 protocol-specific events. Note that buffer-not-ready, Rx buffer and Rx frame events are not affected by this mask. 0 Disable AAL2 receive error events. 1 Enable AAL2 error events.
	15	PM	Enable performance monitoring 0 No performance monitoring for this VC. 1 Perform performance monitoring for this VC. Whenever a cell is received by this VC, the associated performance monitoring table is updated.

NOTE: Entries in boldface must be initialized by the user.

37.4.4.2 CID Mapping Tables and their RxQDs

Each PHY | VP | VC | CID combination is assigned an RxQD using a CID mapping table. To multiplex several receive CIDs into a single common queue, map each multiplexed PHY | VP | VC | CID combination to one RxQD.

The ATM channel's RCT contains the base address of the associated CID mapping table. This base address is external (32 bits) when RCT[MAP] = 1; otherwise, the table resides in the dual-port RAM and the base address is two bytes. The CID of the received packet is used as an index into the mapping table. The mapping table entries are 2-byte RxQD offsets. IF the CID mapping table is external, it must be on the same bus as the BDs and interrupt queues as specified by RCT[BIB].

Offsets between 0–511 belong to the 2048-byte internal RxQD table. It is recommended to have as many RxQDs as possible in the internal table. Note that the first 32 bytes of the internal RxQD table are reserved for internal use; that is, RxQD offsets between 0–7 are reserved. The address of an internal RxQD is RxQD_Base_Int + 4 x RxQD_Offset.

Offsets between 512–65535 belong to the external RxQD table. The address of an external RxQD is RxQD_Base_Ext + 4 x RxQD_Offset. The external RxQD table must be on the same bus as the BDs and interrupt queues as specified by RCT[BIB].

Because the three kinds of RxQDs are each a different size (for example, an SSSAR RxQD is 32 bytes and a CPS switch RxQD is only 4 bytes), some of the offset numbers are left unused.

37.4.4.3 CPS Rx Queue Descriptors

The CPS RxQDs are grouped into RxQDs Tables (one for internal queues and one for external). Each CPS RxQD, as shown in Figure 37-17, points to an CPS RxBD table.

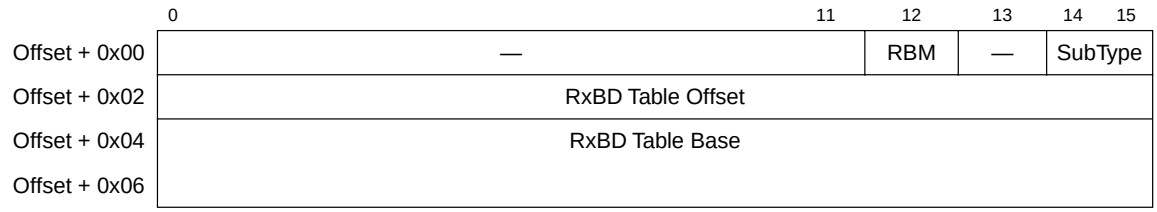


Figure 37-17. CPS Rx Queue Descriptor

Table 37-7 describes the CPS RxQD fields.

Table 37-7. CPS RxQD Field Descriptions

Offset	Bits	Name	Description
0x00	0–11	—	Reserved
	12	RBM	Receive buffer mask. 0 Disable receive buffer interrupt. 1 Enable receive buffer interrupt.
	14–15	SubType	SubType. Sublayer type, should be 00 (CPS) for this descriptor. 00 CPS sublayer. 01 CPS switched. 10 SSSAR. 11 Reserved.
0x02	—	RxBD Table Offset	Holds the offset to the next BD to be opened by the receiver. Should be cleared during initialization.
0x04	—	RxBD Table Base	Holds the pointer to the first BD in the BD table.
NOTE: All fields in this table must be initialized by the user.			

37.4.4.4 CPS Switch Rx Queue Descriptor

The switch RxQD, shown in Figure 37-18, is used for CIDs that are being switched from one $PHY_1 \mid VP_1 \mid VC_1 \mid CID_1$ to another $PHY_2 \mid VP_2 \mid VC_2 \mid CID_2$. The RxQD contains the pointer to the TxQD that controls the TxBD table through which the packet is transferred.

The switch RxQD also contains the translation CID that is saved with the packet in the transmit buffer. A PPD mode enables the discarding of the rest of an SSSAR frame when a buffer is not available.

Note that the CPS switch RxQD must be unique for every Rx switched CID.



Figure 37-18. CPS Switch Rx Queue Descriptor

Table 37-8 describes the CPS switch RxQD fields.

Table 37-8. CPS Switch RxQD Field Descriptions

Offset	Bits	Name	Description
0x00	0–7	TX CID	Translation CID. The received CID is saved in a TX Queue with this new CID number.
	8–12	—	Reserved, should be cleared during initialization.
	12	RBM	Receive buffer mask 0 Disable receive buffer interrupt 1 Enable receive buffer interrupt
	13	PPD	Partial packet discard 0 Normal mode 1 When a buffer-not-ready event causes a packet to be discarded, the remainder of the SSSAR SDU is also discarded. This allows for better performance for switched channels that implement SSSAR.
	14–15	SubType	Sublayer type. Should be 01 (CPS switched) for this descriptor. 00 CPS sublayer 01 CPS switched 10 SSSAR 11 Reserved
0x02	—	TxQD Pointer	Points to the TxQD into which the packets of this CID are stored and later sent.
NOTE: All fields in this table must be initialized by the user.			

37.4.4.5 CPS Receive Buffer Descriptor (RxBD)

The CPS buffer structure consists of a BD table that points to data buffers. The RxBDs contain, apart from the buffer pointer, the packet header, as shown in Figure 37-19. The buffers contain the packet payload.

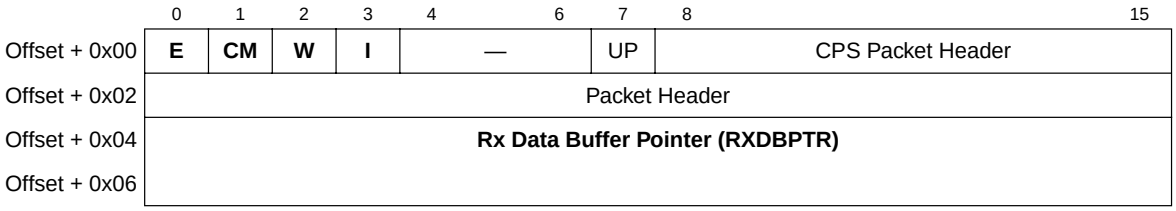


Figure 37-19. CPS Receive Buffer Descriptor

Table 37-9 describes the CPS RxBD fields.

Table 37-9. CPS RxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	**E**	Buffer empty bit 0 The CPS RX buffer is full or data reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP does not use this BD while E remains zero. 1 The CPS RX buffer is empty or reception is in progress. This is controlled by the CP. Once E is set, the core should not access any fields of this buffer.
	1	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear E after this BD is closed, allowing the associated buffer to be reused automatically when the CP next accesses this BD. However, the E bit is cleared if an error occurs while receiving, regardless of the CM bit setting.
	2	**W**	Wrap (final BD in table) 0 This is not the last BD in the RxBD table. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the CP receives incoming data for this channel into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 The CP will not issue an interrupt after this buffer is serviced. 1 The CP will issue an interrupt after this buffer is serviced if the RBM bit in the RxQD is set.
	4–6	—	Reserved
	7	UP	Uncompleted packet 0 No error occurred in this packet 1 A receive error occurred that caused this packet to be uncompleted. The receive error type is reported to the interrupt queue.
	8–15	CPS Packet Header	Contains the beginning of the packet header. See Figure 37-10 for the CPS packet header format.
0x02		CPS Packet Header	Contains the rest of the packet header. The CP checks the packet HEC and if appropriate, indicates a packet HEC error in an interrupt queue entry with CID = 0. See Figure 37-10 for the CPS packet header format.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the CP.

NOTE: Entries in boldface must be initialized by the user.

37.4.4.6 SSSAR Rx Queue Descriptor

The SSSAR RxQD, as shown in Figure 37-20, points to the RxBD table and contains other parameters specific to the SSSAR sublayer. This descriptor can belong to only one PHY | VP | VC | CID.

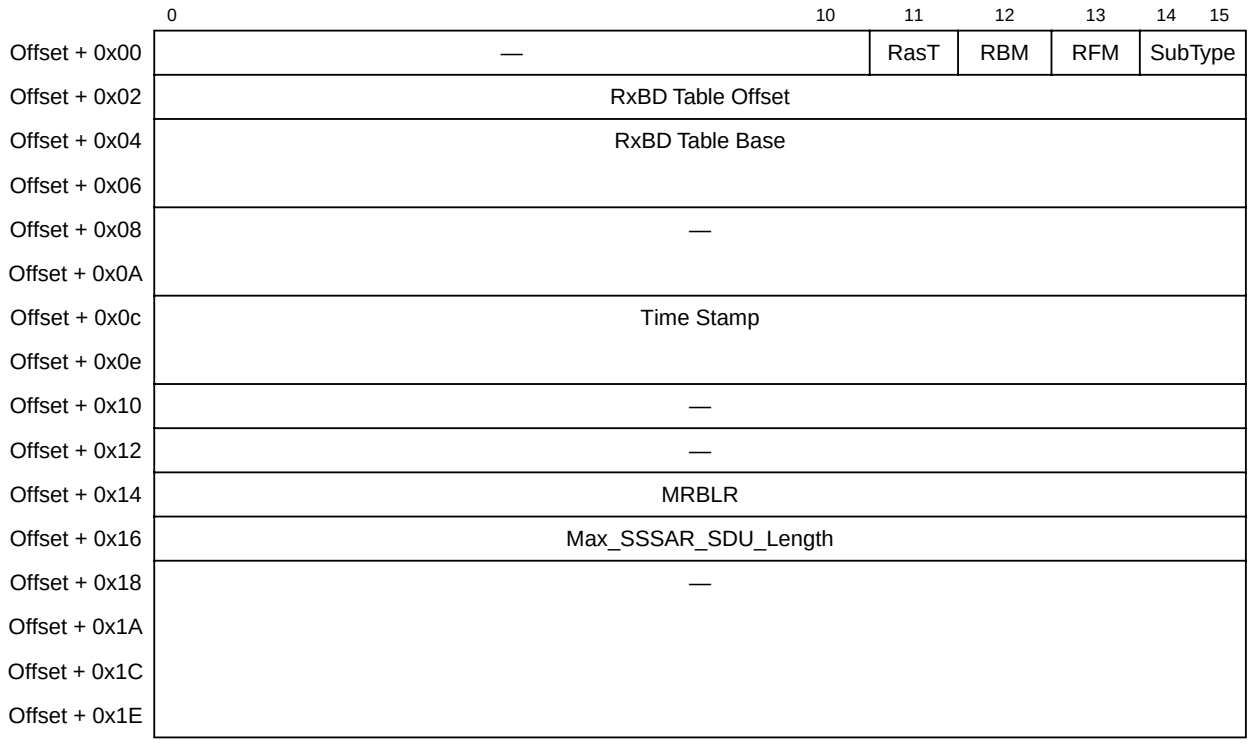


Figure 37-20. SSSAR Rx Queue Descriptor

Table 37-10 describes the SSSAR RxQD fields.

Table 37-10. SSSAR RxQD Field Descriptions

Offset	Bits	Name	Description
0x00	0–11	—	Reserved¥
	11	RasT	Ras Timer enable. 0 Ras Timer disabled (Time Stamp field is still valid) 1 Ras Timer enabled. The Ras Timer duration is set by the Ras Timer Duration parameter in the parameter RAM. If the current SSSAR SDU is not completed before the RasTimer expires, the BD is closed showing the Ras_Timer expired (TE) (SSSAR RxBD[RxError] = 01) and the next packet starts a new SDU.
	12	RBM	Receive buffer mask. 0 Disable receive buffer interrupt 1 Enable receive buffer interrupt
	13	RFM	Receive frame mask. 0 Disable receive frame interrupt 1 Enable receive frame interrupt
	14–15	SubType	Sublayer type. Should be 10 (SSSAR) for this descriptor. 00 CPS sublayer 01 CPS switched 10 SSSAR 11 Reserved

Table 37-10. SSSAR RxQD Field Descriptions (continued)

Offset	Bits	Name	Description
0x02	0–x4	RxBD Table Offset	Points to the next BD to be handled by the CP. The user should initialize this pointer to zero.
0x04	—	RxBD Table Base	Points to the beginning of the BD table.
0x08	—	—	Reserved, should be cleared during initialization.
0x0c		Time Stamp	Used for reassembly timeout of the SSSAR SDU. Whenever the first packet of an SSSAR SDU arrives the timestamp timer is sampled and stored here (regardless of the RasT bit).
0x14	—	MRBLR	Maximum receive buffer length. Holds the maximum receive buffer length. The actual buffer size can be less.
0x16	—	Max_SSSAR_SDU_Length	Holds the maximum SSSAR SDU length. Upon each new packet the accumulated frame size is compared with this value. If the limit is exceeded, the CP discards the rest of the packets of the current frame.

NOTE: All fields except Time Stamp must be initialized by the user.

37.4.4.7 SSSAR Receive Buffer Descriptor

The SSSAR SDU is stored in a BD-buffer structure similar to the structures used for AAL5 frames. The buffer size is determined by SSSAR RxQD[MRBLR]; the actual buffer space used may be smaller. If the received SSSAR SDU is greater than MRBLR, the SDU spans over multiple buffers.

The SSSAR RxBD is shown in Figure 37-21.

	0	1	2	3	4	5	6	7	10	11	15
Offset + 0x00	**E**	**CM**	**W**	**I**	L	**RxError**		—		UUI	
Offset + 0x02	Data Length (DL)										
Offset + 0x04	**Rx Data Buffer Pointer (RXDBPTR)**										
Offset + 0x06											

Figure 37-21. SSSAR Receive Buffer Descriptor

Table 37-11 describes the SSSAR RxBD fields.

Table 37-11. SSSAR RxBD Field Descriptions

Offset	Bits	Name	Description
0x00	0	**E**	Empty 0 The buffer associated with this RxBD is full or data reception was aborted due to an error. The core can read or write any fields of this RxBD. The CP does not use this BD again while E remains zero. 1 The buffer associated with this RxBD is empty or reception is in progress. This RxBD and its receive buffer are controlled by the CP. Once E is set, the core should not write any fields of this RxBD.
	1	**CM**	Continuous mode 0 Normal operation 1 The CP does not clear E after this BD is closed, allowing the associated buffer to be reused automatically when the CP next accesses this BD. However, the E bit is cleared if an error occurs while receiving, regardless of the CM bit setting.
	2	**W**	Wrap (final BD in the table) 0 This is not the last BD in the RxBD table of the current channel. 1 This is the last BD in the RxBD table of this current channel. After this buffer has been used, the CP receives incoming data for this channel into the first BD in the table. The number of RxBDs in this table is programmable and is determined only by the W bit. The current table cannot exceed 64 Kbytes.
	3	**I**	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 An Rx buffer event is sent (provided that RxQD[RBM] is set) to the interrupt queue after this buffer is serviced. The GHIN/GLIN bit in the event register is set when the INT_CNT counter reaches terminal count.
	4	L	Last. Set by the CP. 0 This is not the last buffer of the SSSAR SDU. 1 This is the last buffer of the SSSAR SDU.
	5–6	**RxError**	Rx error occurred 00 No Rx error occurred 01 TE—Ras_Timer expired. The Ras Timer expired before this buffer could be completed. The SSSAR SDU stored in this buffer is not completed. [1] 10 US—Uncompleted SDU. A receive error caused a packet belonging to this SSSAR SDU to be lost. The receiver discarded the rest of this SSSAR SDU. 11 OS—Oversized. The size of the SSSAR SDU has exceeded the Max_SSSAR_SDU_Size parameter. The rest of the SDU was discarded.
	7–10	—	Reserved, should be cleared during initialization.
	11–15	UUI	Contains the UUI of the last packet in the received SDU. Valid only where the L bit is set.
0x02		Data Length	Contains the length of the buffer associated with this BD. If this is the last buffer (L = 1) of the SSSAR SDU, this field contains the total frame length.
0x04	—	**RXDBPTR**	Rx data buffer pointer. Points to the address of the associated buffer. There are no byte-alignment requirements for the buffer, and it may reside in either internal or external memory. This value is not modified by the CP.

[1] When RAS timer expires the RxBD is closed with RAS timer expired indication, the Last (L) bit is not set, and RXF interrupt is not issued. A new RxBD is opened for the next incoming AAL2 packet and the frame is processed as normal and is treated as a new frame. When the next SSSAR end-of-frame indication is received, the RxBD at that time is closed with an L indication, and if RxQD[RFM] = 1, the receiver issues an RXF interrupt.

NOTE: Fields shown in boldface must be user initialized.

37.5 AAL2 Parameter RAM

When configured for ATM mode, the FCC parameter RAM is mapped as shown in Table 37-12. The table includes both the fields for general ATM operation and also the fields specific to AAL2 operation.

Note that some of the values must be initialized by the user, but values updated by the CP should not be modified by the user.

Table 37-12. AAL2 Parameter RAM

Offset	Name	Width	Description
0x00–0x3F	—	—	Reserved. Should be cleared during initialization.
0x40	RCELL_TMP_BASE	Hword	Rx cell temporary base address. Points to a total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64 byte aligned. User-defined. (Recommended address space: 0x3000–0x4000 or 0xB000–0xC000.)
0x42	TCELL_TMP_BASE	Hword	Tx cell temporary base address. Points to total of 64 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User-defined. (Recommended address space: 0x3000–0x4000 or 0xB000–0xC000.)
0x44	UDC_TMP_BASE	Hword	UDC mode only. Points to a total of 32 bytes reserved dual-port RAM area used by the CP. Should be 64-byte aligned. User-defined. (Recommended address space: 0x3000–0x4000 or 0xB000–0xC000.)
0x46	INT_RCT_BASE	Hword	Internal receive connection table base. User-defined.
0x48	INT_TCT_BASE	Hword	Internal transmit connection table base. User-defined.
0x4A	INT_TCTE_BASE	Hword	Internal transmit connection table extension base. User-defined.
0x4C	RAS_Timer_Duration	Word	Contains the RAS_Timer duration in microseconds for the SSSAR sublayer. User-defined.
0x50	EXT_RCT_BASE	Word	External receive connection table base. User-defined.
0x54	EXT_TCT_BASE	Word	External transmit connection table base. User-defined.
0x58	EXT_TCTE_BASE	Word	External transmit connection table extension base. User-defined.
0x5C	UEAD_OFFSET	Hword	User-defined cells mode only. The offset of the UEAD entry in the UDC extra header. Should be an even address. If RCT[BO] = 01 UEAD_OFFSET should be in little-endian format. For example if UEAD entry is the first half word of the extra header in external memory, UEAD_OFFSET should be set to 2 (second half word entry in internal RAM).
0x5E	RxQD_Base_Int	Hword	Points to the base address of the internal RxQD table. The pointer should be 32-byte aligned. User-defined.
0x60	PMT_BASE	Hword	Performance monitoring table base. User-defined.
0x62	APCP_BASE	Hword	APC parameters table base address. User-defined.
0x64	FBT_BASE	Hword	Free buffer pool parameters table base. User-defined.
0x66	INTT_BASE	Hword	Interrupt queue parameters table base. User-defined.

Table 37-12. AAL2 Parameter RAM (continued)

Offset	Name	Width	Description
0x68	—	—	Reserved. Should be cleared during initialization.
0x6A	UNI_STATT_BASE	Hword	UNI statistics table base. User-defined.
0x6C	BD_BASE_EXT	Word	BD table base address extension. BD_BASE_EXT[0–7] hold the 8 most significant bits of the RX/TxBD table base address. BD_BASE_EXT[8–31] should be zero. User-defined.
0x70	VPT_BASE / EXT_CAM_BASE	Word	Base address of the address compression VP table/external CAM. User-defined.
0x74	VCT_BASE	Word	Base address of the address compression VC table. User-defined.
0x78	VPT1_BASE / EXT_CAM1_BASE	Word	Base address of the address compression VP1 table/EXT CAM1. User-defined.
0x7C	VCT1_BASE	Word	Base address of the address compression VC1 table. User-defined.
0x80	VP_MASK	Hword	VP mask for address compression lookup. User-defined.
0x82	VCI_Filtering	Hword	VCI filtering enable bits. When cells with VCI = 3, 4, 6, 7–15 are received and the associated VCI_Filtering bit = 1 the cell is sent to the raw cell queue. VCI = 3 is associated with VCI_Filtering[3], VCI = 15 is associated with VCI_Filtering[15]. VCI_Filtering[0–2, 5] should be zero. See Section 35.10.1.2, “VCI Filtering (VCIF),” in the *MPC8560 User’s Manual*.
0x84	GMODE	Hword	Global mode. User-defined. See Section 35.10.1.3, “Global Mode Entry (GMODE),” in the *MPC8560 User’s Manual.*
0x86	COMM_INFO	Hword	The information field associated with the last host command. User-defined. See Section 35.14, “ATM Transmit Command,” in the *MPC8560 User’s Manual.*
0x88		Hword	
0x8A		Hword	
0x8C	—	Word	Reserved. Should be cleared during initialization.
0x90	CRC32_PRES	Word	Preset for CRC32. Initialize to 0xFFFFFFFF.
0x94	CRC32_MASK	Word	Constant mask for CRC32. Initialize to 0xDEBB20E3.
0x98	AAL1_SNPT_BASE	Hword	AAL1 SN protection look up table base address. (AAL1 only.) The 32-byte table resides in dual-port RAM and must be initialized by the user (See Section 35.10.6, “AAL1 Sequence Number (SN) Protection Table (AAL1 Only),” in the *MPC8560 User’s Manual*.).
0x9A	—	Hword	Reserved. Should be cleared during initialization.
0x9C	SRTS_BASE	Word	External SRTS logic base address. (AAL1 only.) Should be 16-byte aligned.
0xA0	IDLE/UNASSIGN_BASE	Hword	Idle/unassign cell base address. Points to dual-port RAM area contains idle/unassign cell template (little-endian format). Should be 64-byte aligned. User-defined. The ATM header should be 0x0000_0000 or 0x0100_0000 (CLP = 1).
0xA2	IDLE/UNASSIGN_SIZE	Hword	Idle/Unassign cell size. 52 in regular mode. 53–64 in UDC mode.
0xA4	EPAYLOAD	Word	Reserved payload. Initialize to 0x6A6A6A6A.

Table 37-12. AAL2 Parameter RAM (continued)

Offset	Name	Width	Description
0xA8	Trm	Word	(ABR only) The upper bound on the time between F-RM cells for an active source. TM 4.0 defines the Trm period as 100 msec. The Trm value is defined by the system clock and the time stamp timer prescaler (See RTSCR). For time stamp prescalar of 1µs, Trm should be set to 100 ms/1µs = 100,000.
0xAC	Nrm	Hword	(ABR only) Controls the maximum cells the source may send for each F-RM cell. Set to 32 cells.
0xAE	Mrm	Hword	(ABR only) Controls the bandwidth between F-RM, B-RM and user data cell. Set to 2 cells.
0xB0	TCR	Hword	(ABR only) Tag cell rate. The minimum cell rate allowed for all ABR channels. An ABR channel whose ACR is less than TCR sends only out-of-rate F-RM cells at TCR. Should be set to 10 cells/sec as defined in the TM 4.0. Uses the ATMF TM 4.0 floating-point format. Note that the APC minimum cell rate should be at least TCR.
0xB2	ABR_RX_TCTE	Hword	(ABR only) Points to total of 16 bytes reserved dual-port RAM area used by the CP. Should be 16-byte aligned. User-defined.
0xB4	RxQD_Base_Ext	Word	Points to the base address of the external RxQD table. The actual address of the first RxQD in the table is RxQD_Base_Ext + 512 x 4. User-defined.
0xB8	RX_UDC_Base	Word	Valid only for AAL2 VCs. Points to the base of the RX UDC header table that contains the UDC headers of the AAL2 VCs. The pointer to a VC UDC header is: RX_UDC_Base + 16 x CH# (where CH# is the ATM channel number)
0xBC	TX_UDC_Base	Word	Valid only for AAL2 VCs. Points to the base of the TX UDC header table that contains the UDC headers of the AAL2 VCs. The pointer to a VC UDC header is: TX_UDC_Base + 16 x CH# (where CH# is the ATM channel number)
0xC0–0xDF	—	—	Reserved. Should be cleared during initialization.
0xE0	TCELL_TMP_BASE_EXT	Word	Transmit Cell Temporary base address. Points to a total of 64 x last_AAL2_Ch# octets reserved in external memory for partially filled cells. Note: TCELL_TMP_BASE_EXT must be on the same bus as the all the AAL2 data buffers required for CPS, SSSAR and CID switching. (InCell mode)
0xE4–0xFB	—	—	Reserved. Should be cleared during initialization.
0xFC	PAD_TMP_BASE	Hword	PAD template base address. Points to an internal memory area that contains the zero cell template. Should be 64-byte aligned. User-defined.
0xFE	—	—	Reserved. Should be cleared during initialization.

37.6 User Defined Cells in AAL2

The user defined cell (UDC) mode for ATM is described in Section 36.7, “User-Defined Cells (UDC),” in the *MPC8560 User’s Manual*, which applies generally to AAL2 operation

as well. However, for AAL2 operation only, the UDC headers reside in a table in external memory, not in the BDs.

For transmit channels in AAL2 UDC mode, initialize its UDC header entry in the TX UDC header table before activating the channel. The header can be up to 12 bytes. The TX_UDC_Base parameter in the parameter RAM (see Table 37-12), points to the beginning of the TX UDC header table.

The UDC header of a specific AAL2 transmit VC is located at the following address:

TX_UDC_Base + CH# x 16 (where CH# is the ATM channel number)

For receive channels in AAL2 UDC mode, the receiver copies the UDC header from the first cell received by the VC to the RX_UDC header table. The UDC headers of subsequent cells of that VC are discarded; UDC extended address mode (UEAD) is not affected.

The UDC header of a specific AAL2 receive VC is located at the following address:

RX_UDC_Base + CH# x 16 (where CH# is the ATM channel number)

The structure of a UDC header table (receive or transmit) is shown in Figure 37-22.

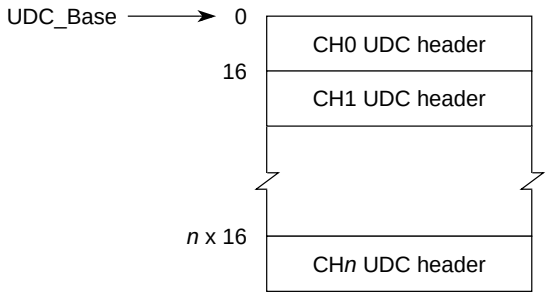


Figure 37-22. UDC Header Table

37.7 AAL2 Exceptions

For each VC, four circular interrupt queues are available. By programming RCT[INTQ] and TCT[INTQ] for each VC, the user assigns an interrupt queue number.

When one of the CIDs generates an interrupt request, the CP writes a new entry to the interrupt queue containing the ATM channel number, the CID and a description of the exception. Because CID = 0 is a unique CID number, it is used to specify that the event is related to the VC rather than the CID. As with all ATM exceptions, the valid (V) bit is then set and INTQ_PTR is incremented. When INTQ_PTR reaches a location with the W bit set, it wraps to the first entry in the queue. More details can be found in Section 35.11, “ATM Exceptions.”

An interrupt entry for a CID is shown in Figure 37-23.

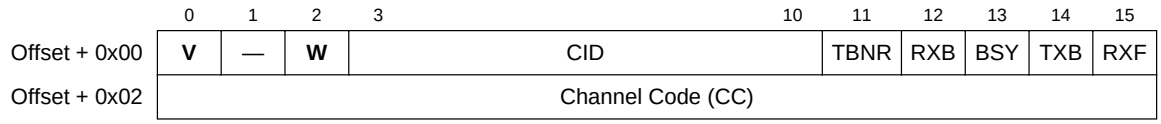


Figure 37-23. AAL2 Interrupt Queue Entry CID ≠ 0

Table 37-13 describes the interrupt queue entry fields for a CID.

Table 37-13. AAL2 Interrupt Queue Entry CID ≠ 0 Field Descriptions

Offset	Bits	Name	Description
0x00	0	**V**	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the CP. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	—	—
	2	**W**	Wrap bit. When set, this is the last interrupt entry in the circular table. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	CID	CID number. The exception occurred for this CID.
	11	TBNR	Tx buffer not ready interrupt. This interrupt is issued when the CP tries to open a TxBD, which is not ready (R = 0). This interrupt is sent only if TxQD[BNM] = 1. The interrupt has an associated channel code and CID.
	12	RXB [1]	Rx buffer interrupt. This interrupt is issued when the I bit is set for an RxBD and the RxQD[RBM] bit is set. This interrupt has an associated channel code and CID.
	13	BSY	Busy condition. The RxBD table associated with this channel's CID is busy. Packets were discarded due to this condition.
	14	TXB	Transmit buffer interrupt. This interrupt is issued when the TxBD[I] bit is set. This interrupt is sent only if TxQD[TBM] is set. This interrupt has an associated channel code and CID.
	15	RXF[1]	Receive SSSAR SDU (frame). An SSSAR frame belonging to this channel's CID has been received. This interrupt is sent only if RxQD[RFM] = 1.
0x02	—	CC	Channel code specifies the ATM channel number associated with this interrupt.

[1] These interrupt queue fields are defined differently for other AAL types. Refer to the *MPC8560 User's Manual* (Table 35-43) for more information.

An interrupt entry for the VC is shown in Figure 37-24.

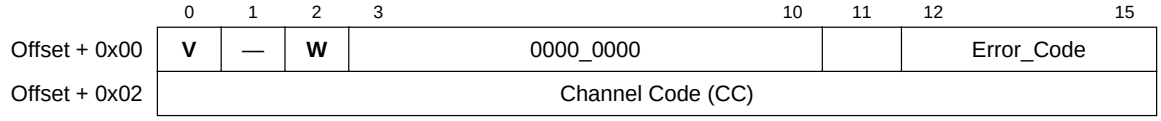


Figure 37-24. AAL2 Interrupt Queue Entry CID = 0

Table 37-14 describes the interrupt queue entry fields for the VC. All the receive error events are enabled by setting RCT[EM].

Table 37-14. AAL2 Interrupt Queue Entry CID = 0 Field Descriptions

Offset	Bits	Name	Description
0x00	0	**V**	Valid interrupt entry 0 This interrupt queue entry is free and can be used by the CP. 1 This interrupt queue entry is valid. The host should read this interrupt and clear this bit.
	1	—	—
	2	**W**	Wrap bit. When set, this is the last interrupt circular table entry. During initialization, the host must clear all W bits in the table except the last one, which must be set.
	3–10	CID	CID number. Equals zero. This exception applies to the whole cell.
	11	—	reserved
	12–15	Error_Code	A receive error was detected. 0000 Parity error of the OSF. 0001 The STF sequence number is incorrect. 0010 The number of octets expected to overlap into this cell does not match the OSF. 0011 OSF is greater than 47. 0100 A packet HEC error was detected. 0101 The length of the CPS packet exceeds the Max_SDU_Length. 0111 A packet HEC error was detected in a split header packet.
0x02	—	CC	Channel code specifies the ATM channel number associated with this interrupt.

Chapter 38
Transmission Convergence Layer

The MPC8560 can support applications which receive ATM traffic over the standard serial protocols like E1, T1, and xDSL via its serial interface (SIx TDMx and NMSI) ports because the ATM transmission convergence (TC) layer functionality is implemented internally. This allows the use of standard low-cost PHY devices in system applications instead of PHYs that support UTOPIA bus devices.

A typical TC layer application requires the use of one SI TDM channel per TC block. As shown in Figure 38-1, all TC blocks are internally connected to FCC2. In addition, Figure 38-1 shows FCC1 connected to a UTOPIA 16-bit MPHY bus which can be routed outside and operated independently of the TC blocks.

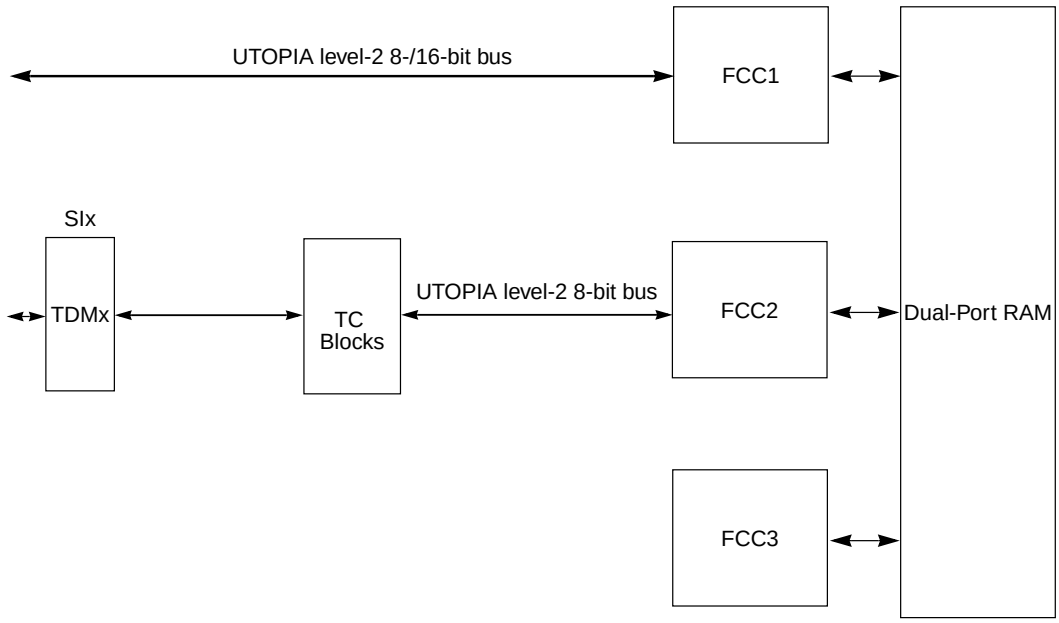


Figure 38-1. Serial ATM Using FCC2 and TC Blocks (Single Channel)

38.1 Features

The major TC layer features are as follows:

- Eight TDM channels routed in hardware to a TC layer block.
  - — Protocol-specific overhead bits may be discarded or routed to other controllers by the SI performing ATM TC layer functions (according to ITU-T I.432)
  - — Transmit (Tx) updates are as follows:
    - – Cell HEC generation
    - – Payload scrambling using self synchronizing scrambler (programmable by the user)
    - – Coset generation (programmable by the user)
    - – Cell rate by inserting idle/unassigned cells
  - — Receive (Rx) updates are as follows:
    - – Cell delineation using bit by bit HEC checking and programmable ALPHA and DELTA parameters for the delineation state machine
    - – Payload descrambling using self synchronizing scrambler (programmable by the user)
    - – Coset removing (programmable by the user)
    - – Filtering idle/unassigned cells (programmable by the user)
    - – Performing HEC error detection and single bit error correction (programmable by the user)
    - – Generating loss of cell delineation status/interrupt
- Operates with FCC2 (UTOPIA 8)
- Serial loop back mode
- Cell echo mode
- Supports both FCC transmit modes:
  - — External rate mode—Idle cells are generated by the FCC (microcode) to control data rate
  - — Internal rate mode (sub-rate)—FCC transfers only the data cells using the required data rate. The TC layer generates idle/unassigned cells to maintain the line bit rate
- Supports the TC layer and PMD (physical medium dependant) WIRE interface (according to the ATM-Forum af-phy-0063.000)
- Cell counters for performance monitoring:
  - — 16-bit counters count:
    - – HEC errorred cells
    - – HEC single bit errorred and corrected cells
    - – Idle/unassigned cells filtered
    - – Idle/unassigned cells transmitted

  - Transmitted ATM cells
  - Received ATM cells
  - Maskable interrupt is sent to the host when a counter expires.
- Overrun (Rx cell FIFO) and underrun (Tx cell FIFO) condition produces maskable interrupt
- May be operated at E1 and DS-1 rates. In addition, xDSL applications at bit rates up to 10 Mbps are supported.

38.2 TC Layer Block Diagram

The TC layer block is described in Figure 38-2. The transmit and the receive parts are independent; the only case in which they are synchronized is in cell echo mode.

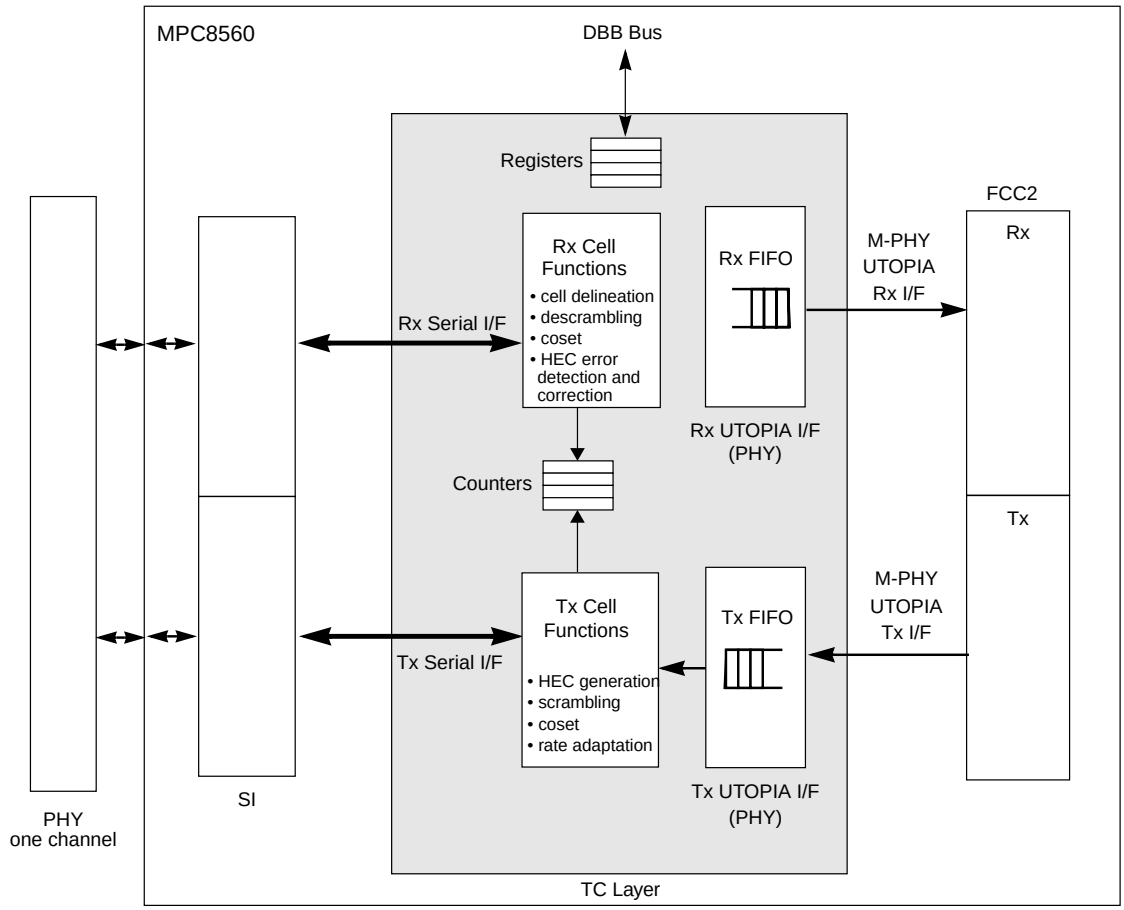


Figure 38-2. TC Layer Block Diagram

38.3 Signals

The TC layer is operated via an SI TDM port or NMSI port using a serial protocol. Synchronization signals are required for some applications and must be supported. Table 38-1 describes the signals required for operating the TC layer.

Table 38-1. TC Layer Signals

Signal	Direction	Description
Txc	Input	Tx clock. Clocks Tx data out of the TC to external device.
Txd	Output	Tx data from TC to external device.
Txsyn	Input	Tx synch. Synchronizes the transmit data to the beginning of a frame.
Rxc	Input	Rx clock. Clocks the data into the TC.
Rxd	Input	Rx data. From external device to the TC.
Rxsyn	Input	Rx synch. Synchronizes the received data. Not required in NMSI mode.

38.4 Receive ATM Cell Functions

The ATM receive cell functions block (RCF) performs the receive functions of the TC block. It performs cell delineation, cell payload descrambling, HEC verification and correction, and idle/unassigned cell filtering.

Cell delineation is the process of framing data to ATM cell boundaries using the header error check (HEC) received in the ATM cell header. The HEC is a CRC-8 calculation over the first four octets of the ATM cell header. The cell delineation algorithm assumes that repetitive correct HEC calculations over consecutive cells indicate valid ATM cell boundaries.

The RCF performs a sequential bit by bit hunt for a correct HEC sequence. While performing this hunt, the cell delineation state machine is in HUNT state. When a correct HEC is found, the RCF locks on the particular cell boundary and enters the PRESYNCH state, which indicates that the previously detected HEC pattern is not a false indication. If a correct HEC pattern is false, an incorrect HEC is received within the next DELTA cells. If an incorrect cell is detected, then a transition to the HUNT state is made. If an incorrect HEC is not detected in the PRESYNCH state, a transition to the SYNCH state is made. In the SYNCH state, the TC is assumed to be synchronized so that other functions can be applied to the received cell. A transition back to the HUNT state is made only after ALPHA consecutive incorrect HEC patterns are detected.

The cell delineation state machine is shown in Figure 38-3. The ALPHA and DELTA parameters determine the robustness of the delineation method. ALPHA determines the robustness against false misalignment due to bit errors. DELTA determines the robustness against false delineation in the synchronization. Both parameters are programmable for

each TC block and are provided to help tune the system according to the line error characteristics of a specific application.

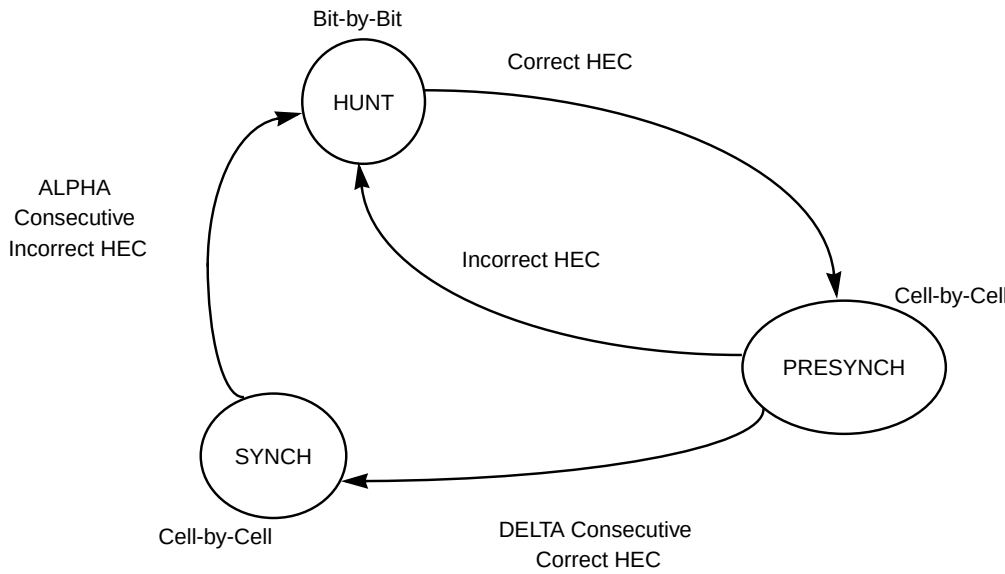


Figure 38-3. TC Cell Delineation State Machine

The RCF descrambles (programmable) the cell payload using the self-synchronizing descrambler with a polynomial of $x^{43} + 1$.

The HEC calculation is a CRC-8 calculation over the first four octets of the ATM cell header. The RCF verifies the received HEC using the accumulation polynomial, $x^8 + x^2 + x + 1$. The coset polynomial $x^6 + x^4 + x^2 + 1$ is added (modulo 2) to the received HEC octet before comparison with the calculated result (programmable).

The RCF can perform single bit error correction on the header. If multiple bit errors are found in the HEC, the cell is discarded. If the single bit error correction mode is not enabled (TCMODE[SBC] = 1), the cell is also discarded when a single bit error is found in the header.

When the cell delineation state machine is in the SYNCH state, the HEC verification state machine (see Figure 38-4) implements the correction algorithm. This state machine makes sure that a single cell header is corrected at a time. If consecutive cells are detected with single bit errors in their headers, only the first cell error is corrected and the rest are discarded. This state machine reduces the probability of the delivery of cells with errorred headers under bursty error conditions.

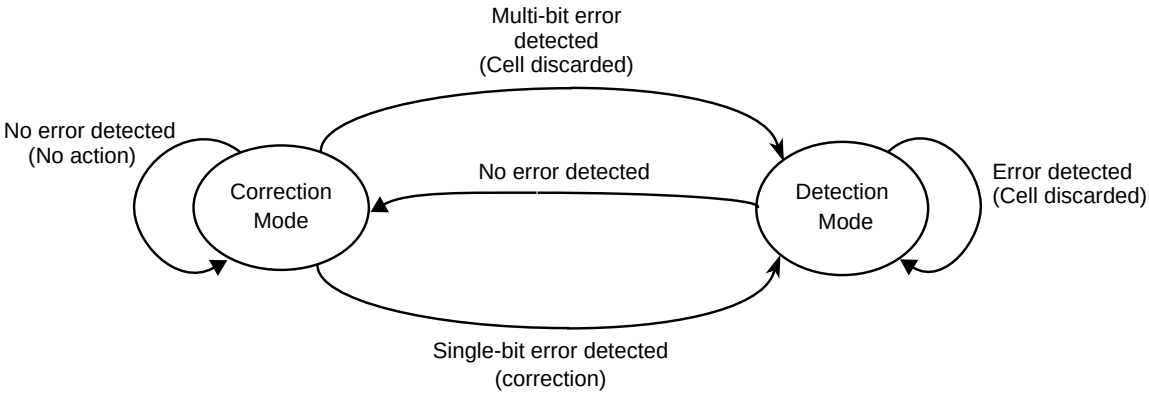


Figure 38-4. HEC: Receiver Modes of Operation

The RCF can also perform idle/unassigned cell filtering. Both features are programmable (TCMODE[CF]). Cells that are detected to be idle/unassigned are discarded, that is, not forwarded to the UTOPIA interface Rx FIFO.

38.5 Receive ATM 2-Cell FIFO

The receive FIFO provides FIFO management and an interface to the UTOPIA receive cell interface. The receive FIFO can hold 2 ATM cells, thereby providing the cell rate decoupling function between the transmission system physical layer and the ATM layer.

FIFO management includes filling the FIFO, indicating to the UTOPIA interface that it contains cells, maintaining the FIFO read and write pointers, and detecting FIFO overrun (TCER[OR]) conditions.

38.6 Transmit ATM Cell Functions

The transmit ATM cell functions block (TCF) performs the ATM cell payload scrambling and is responsible for the HEC generation and the idle/unassigned cell generation.

The TCF scrambles (programmable by the user) the cell payload using the self-synchronizing scrambler with polynomial $x^{43} + 1$.

The HEC is generated using the polynomial $x^8 + x^2 + x + 1$. The coset polynomial $x^6 + x^4 + x^2 + 1$ is added (modulo 2) (programmable by the user) to the calculated HEC octet. The result overwrites the HEC octet on the transmitted cell. When the transmit FIFO is empty, the TCF inserts idle/unassigned cells (counted in ICC).

The TCF accumulates the number of transmitted assigned cells in a counter (TCC).

38.7 Transmit ATM 2-Cell FIFO

The transmit FIFO provides FIFO management and an interface to the UTOPIA transmit interface. The FIFO provides the cell rate decoupling between the transmission system physical layer and the ATM layer.

The FIFO management includes emptying cells from the transmit FIFO, indicating to the UTOPIA interface that it is full, maintaining the FIFO read and write pointers, and detecting FIFO underrun (TCER[UR]) conditions.

38.8 Rx UTOPIA Interface

This block performs the receive interface with the FCC via the UTOPIA bus. It implements the UTOPIA level-2 (multi-PHY) 8-bit PMD side (slave) interface.

38.9 Tx UTOPIA Interface

This block performs the transmit interface with the FCC via the UTOPIA bus. It implements the UTOPIA level-2 (multi-PHY) 8-bit PMD side (slave) interface.

38.10 TC Layer Connection to FCC2

The FCC2 is connected to the TC layer using the FC2 field of the CMX FCC clock route register (CMXFCR).

	0	1	2 4	5 7	8 9	10 12	13 15
Field	—	FC1	RF1CS	TF1CS	FC2	RF2CS	TF2CS
Reset	0000_0000_0000_0000						
R/W	R/W						
Addr	0x9_1B04						

	16	17	18 20	21 23	24 31
Field	—	FC3	RF3CS	TF3CS	—
Reset	0000_0000_0000_0000				
R/W	R/W				
Addr	0x9_1B06				

Figure 38-5. CMX FCC Clock Route Register (CMXFCR)

Table 38-2 describes the CMXFCR[FC2] field.

Table 38-2. CMXFCR[FC2] Field Description

Bits	Name	Description
8-9	FC2	Defines the FCC2 connection. 00 FCC2 is not connected to the TSA and is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus FCCn pins is made in the parallel I/O control register. 01 FCC2 is connected to the TSA of the SIs. The NMSIx pins are available for other purposes. 10 FCC2 is connected to the TC layer. The NMSIx pins are available for other purposes. 11 Reserved

38.11 TC Layer Programming Model

This section describes the TC layer-specific registers and other programming model features.

38.11.1 TC Layer Registers

Each TC layer block is controlled by registers located in the block and accessed from the system bus.

38.11.1.1 TC Layer Mode Register (TCMODE)

Each TC layer block is configured using a TC layer mode register TCMODEx, as shown in Figure 38-6.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	RXEN	TXEN	RPS	TPS	RC	TC	SBC	CF		URE	LB		TBA	IMA	SM	CM
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W															

Figure 38-6. TCMODEx

Table 38-3 describes TCMODE fields.

Table 38-3. TCMODE Field Descriptions

Bits	Name	Description
0	RXEN	TC Layer Rx enable bit. Enables the TC Layer Rx block operation: 0 TC Layer Rx operation is disabled. 1 TC Layer Rx operation is enabled.
1	TXEN	TC Layer Tx enable bit. Enables the TC Layer Tx block operation: 0 TC Layer Tx operation is disabled. 1 TC Layer Tx operation is enabled.
2	RPS	Rx Payload DeScrambling 0 Payload descrambling is performed on received payload data. 1 No payload descrambling is performed on received payload data.

Table 38-3. TCMODE Field Descriptions (continued)

Bits	Name	Description
3	TPS	Tx Payload Scrambling 0 Payload scrambling is performed on transmitted payload data. 1 No payload scrambling is performed on transmitted payload data.
4	RC	Rx Coset Enable 0 XOR with 0xAA is done on received HEC. 1 No XOR with 0xAA is done on received HEC.
5	TC	Tx Coset Enable 0 XOR with 0xAA is done on transmitted HEC. 1 No XOR with 0xAA is done on transmitted HEC.
6	SBC	Header Single Bit error Correction 0 Perform single bit error correction on the header according to HEC while in Synch mode. 1 Do not perform single bit error correction on the header.
7–8	CF	Rx Idle/unassigned Cells Filtering 00 No cell filtering is done on Rx cells. 01 Idle cell filtering is done - idle cells are discarded. 10 Unassigned cell filtering is done - unassigned cells are discarded. 11 Idle and unassigned cell filtering is done - both idle and unassigned cells are discarded. The Header of idle cell (ITU-T I.361): b00000000_00000000_00000000_00000001 The Header of unassigned cell (ITU-T I.361): b00000000_00000000_00000000_0000xxx0 Note that physical layer cells bypass the TC layer; they are not filtered. Also note that the filter works on the header only and ignores the HEC.
9	URE	Underrun interrupt (TCER[UR]) enable. Underrun interrupt may be set when Idle cell is generated by the TC. 0 Underrun interrupt disabled. 1 Underrun interrupt enabled.
10–11	LB	Loopback/echo modes 00 Normal operation. 01 Cell echo mode operation. Received cells are transmitted and do not go out to the UTOPIA bus. 10 Data loopback mode operation. Transmit data stream is connected to the receive data stream. 11 Not used. Note that for echo mode operation, TCMODE[SM] should be cleared, independent of the FCC multi-PHY mode configuration.
12	TBA	Tx Byte align 0 Tx data is transferred as soon as it is enabled. 1 Tx data is transferred byte aligned to the Txsyn signal.
13	IMA	IMA mode 0 Rx is not in IMA. 1 Rx is in IMA mode.
14	SM	Single mode 0 TC is not the only PHY on UTOPIA 1 TC is the only PHY on UTOPIA

Table 38-3. TCMODE Field Descriptions (continued)

Bits	Name	Description
15	CM	Cell counters mode 0 Reading a cell counter clears the counter. 1 Reading a cell counter does not change the counter's value.

38.11.1.2 Cell Delineation State Machine Register (CDSMRx)

The cell delineation state machine register (CDSMR), as shown in Figure 38-7, holds the ALPHA and DELTA parameters of the cell delineation state machine.

	0				4	5				9	10					15
Field	ALPHA					DELTA					—					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W															

Figure 38-7. CDSMRx

Table 38-4 describes CDSMR fields.

Table 38-4. CDSMR Field Descriptions

Bits	Name	Description
0-4	ALPHA	ALPHA consecutive received cells with incorrect HEC are counted by the cell delineation state machine to pass from state SYNCH to state HUNT.
5-9	DELTA	DELTA consecutive received cells with correct HEC are counted by the cell delineation state machine to pass from state PRESYNCH to state SYNCH.
10–15	—	Reserved

38.11.1.3 TC Layer Event Register (TCERx)

The TC layer event registers (TCERx), as shown in Figure 38-8, records error events for each TC block. TCER event bits are cleared by writing ones to them.

	0	1	2	3	4	5				9	10	11	12	13	14	15
Field	OR	UR	CDT	MS	PARE	—					ROF	TOF	EOF	COF	IOF	FOF
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W															

Figure 38-8. TCERx

The TCER bits are described in Table 38-5.

Table 38-5. The TCER Field Descriptions

Bits	Name	Description
0	OR	Overrun. Rx FIFO OverFlow. Set when Rx FIFO is full and another complete cell is received. The cell is discarded.
1	UR	Underrun. No ATM cell to transmit. Set when the Tx FIFO is empty and the transmission of a cell is completed. An idle cell is sent. This interrupt is enabled only if TCMODE[URE] is set. The idle cell header is: 0x00000001 (I.432), whose HEC is: 0x52 The idle cell payload is:0x6A (I.432).
2	CDT	Cell delineation toggled. Set when the cell delineation bit (CD) in TCGSR has changed.
3	MS	Misplaced Txsyn signal Set when Txsyn is out of place. (The first Txsyn is by definition always in place.)
4	PARE	Parity event Set when parity from UTOPIA to transmit is wrong.
5-9	—	Reserved
10	ROF	Received cell counter overflow Set when the received cells counter passes its maximum value.
11	TOF	Transmitted cell counter overflow Set when the transmitted cells counter passes its maximum value.
12	EOF	Errorred cells counter overflow Set when the errorred cells counter passes its maximum value.
13	COF	Corrected cells counter overflow Set when the corrected cells counter passes its maximum value.
14	IOF	Tx Idle cells counter overflow Set when the Tx idle cells counter passes its maximum value.
15	FOF	Filtered cells counter overflow Set when the filtered cells counter passes its maximum value.

38.11.1.4 TC Layer Mask Register (TCMRx)

This register's field description is identical to that of TCER (See Section 38.11.1.3, "TC Layer Event Register (TCERx)."). Each bit that is set in TCMR enables an interrupt when the corresponding bit in TCER is set.

38.11.1.5 CPM Low Interrupt Priority Register (SCPRR_L)

The CPM low interrupt priority register (SCPRR_L), shown in Figure 38-9, defines prioritization of the TC layer and the SCCs.

	0–2	3–5	6–8	9–11	12–15
Field	YC1P	YC2P	YC3P	YC4P	—
Reset	000	001	010	011	0000
R/W	R/W				
Addr	0x9_0C18				

	16–18	19–21	22–24	25–27	28–31
Field	YC5P	YC6P	YC7P	YC8P	—
Reset	100	101	110	111	0000
R/W	R/W				
Addr	0x9_0C20				

Figure 38-9. CPM Low Interrupt Priority Register (SCPRR_L)

Table 38-6 describes SCPRR_L fields.

Table 38-6. SCPRR_L Field Descriptions

Bits	Name	Description
0–2	YC1P	Priority order. Defines which SCC/TC layer asserts its request in the YCC1 priority position. Do not program the same SCC to multiple priority positions. This field can be changed dynamically. 000 SCC1 asserts its request in the YCC1 position. 001 SCC2 asserts its request in the YCC1 position. 010 SCC3 asserts its request in the YCC1 position. 011 SCC4 asserts its request in the YCC1 position. 100 TC layer asserts its request in the YCC1 position. All other configurations: YCC1 position is not active.
3–11	YC2P–YC8P	Same as YC1P, but for YCC2–YCC8
12–15	—	Reserved, should be cleared.

38.11.2 TC Layer General Registers

The TC layer general registers are registers that are distributed to all of the TC blocks. Each TC block is represented by specific bits. When accessing a general register each TC block is responsible for its specific bits only.

38.11.2.1 TC Layer General Event Register (TCGER)

The TC layer general event register (TCGER), as shown in Figure 38-10, summarizes the events for all the TC blocks. Each bit stands for an ORed event register of a TC block. Once a bit is set, it indicates that one or more event bits are set in the corresponding TC block event register.

	0	1	2	3	4	5	6	7	8							15
Field	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	—							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W															

Figure 38-10. TCGER

Table 38-7 describes TCGER fields.

Table 38-7. TCGER Field Descriptions

Bits	Name	Description
0	TC1	One bit or more is set in TC1 event register.
1	TC2	One bit or more is set in TC2 event register.
2	TC3	One bit or more is set in TC3 event register.
3	TC4	One bit or more is set in TC4 event register.
4	TC5	One bit or more is set in TC5 event register.
5	TC6	One bit or more is set in TC6 event register.
6	TC7	One bit or more is set in TC7 event register.
7	TC8	One bit or more is set in TC8 event register.

38.11.2.2 TC Layer General Status Register (TCGSR)

Figure 38-11 shows the TC layer general status register (TCGSR), which records the cell delineation and transmit FIFO status for all TC blocks.

	0	1	2	3	4	5	6	7	8							15
Field	CD1	CD2	CD3	CD4	CD5	CD6	CD7	CD8	—							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R															

Figure 38-11. TCGSR

Table 38-8 describes TCGSR fields.

Table 38-8. TCGSR Field Descriptions

Bits	Name	Description
0–7	CDx	Cell Delineation. The cell delineation state machine status of TCx. 0 - Cell delineation state machine is in Hunt or Pre-Synch modes. 1 - Cell delineation machine is in Synch mode.
8–15	—	Reserved

38.11.3 TC Layer Cell Counters

Each TC block maintains six memory-mapped 16-bit performance cell counters that are updated during operation and can be read by the host. If a counter overflows, it wraps back to zero and generates a maskable interrupt. These counters are automatically cleared when read if TCMODE[CM] = 0; see Section 38.11.1.1, “TC Layer Mode Register (TCMODE).”

38.11.3.1 Received Cell Counter (RCC)

This cell counter is updated whenever a received cell without HEC errors is passed to the Rx UTOPIA FIFO.

38.11.3.2 Transmitted Cell Counter (TCC)

This cell counter is updated whenever the transmission of a cell is completed.

38.11.3.3 Errorred Cell Counter (ECC)

This cell counter is updated whenever a received errorred cell (cell with header error) is discarded.

38.11.3.4 Corrected Cell Counter (CCC)

This cell counter is updated whenever a received cell with a HEC single bit error is corrected. If header single bit error correction is not enabled (TCMODE[SBC] is set), this counter is not updated. (All errorred cells are counted by the errorred cell counter (ECC).)

38.11.3.5 Tx IDLE Cell Counter (ICC)

This cell counter is updated whenever an idle cell is transmitted.

38.11.3.6 Filtered Cell Counter (FCC)

This cell counter is updated whenever an idle/unassigned cell is filtered (discarded). If cell filters are not enabled (TCMODE[CF] is cleared), this counter is not updated.

38.11.4 Programming FCC2

FCC2 is designed to work with the TC blocks. The TC blocks are located on fixed addresses on the UTOPIA bus internally. FCC2 should be programmed to work with the TC blocks as if the TC blocks are external PHYs located on the lowest eight (or fewer) addresses.

38.11.5Programming and Operating the TC Layer

The host should first program the mode registers of each TC block to be active, according to the number of TC channels required. Then, FCC2 should be programmed to work on the UTOPIA interface with the active TC blocks. Finally, FCC2 and the PHYs regarding the TC channels are enabled.

The transmit channels for each TC block are enabled by setting TCMODEx[TXEN].

The receive channels for each TC block are enabled by setting TCMODEx[RXEN]. The host polls the CD bits of each enabled TC block to see that its receive cell delineation state machines are synchronized. For every TC block that is synchronized, the host clears the CDT bit in its event register. Once all the enabled TC blocks are synchronized, the host terminates its initialization routine, and the system starts normal operation.

Once a TC block gets out of synchronization, the corresponding TCGSR[CD] is cleared. This change then causes a TCER[CDT] interrupt to the host (if enabled).

On the receive path, the TC receives the bit stream via the SI. The TC first attempts to gain synchronization on the ATM cell boundaries by checking each byte (HEC candidate) against the HEC calculated on the preceding 32 bits (ATM cell header candidate). Once synchronized, it performs the descrambling function on the cell payload (if enabled), performs the coset function on the HEC (if enabled), checks for HEC errors and corrects single HEC errors when found (again, if enabled). Cells containing multi-bit header errors (at least 2 errors) are discarded. Idle and unassigned cells are filtered (discarded) when detected (if the filters are enabled). Once a cell's processing is complete, it is passed to the TC receive FIFO, and the internal TC cell counters are updated. The cell is passed from the TC receive FIFO via the UTOPIA interface to the FCC2 receive FIFO.

An overrun condition occurs when the TC receive FIFO is full and the FCC for some reason (CP is busy) is unable to read a cell from it (via the UTOPIA interface) before another valid cell is received. The incoming cell is discarded and TCER[OR] interrupt is sent to the host (if enabled).

On the transmit path, once enabled, the TC starts requesting for cells to send via the UTOPIA interface. When a cell is passed via the UTOPIA interface to the TC transmit block, it is stored in the TC transmit FIFO. When a cell is to be sent, it is read from the TC transmit FIFO and is processed. The scrambling function is performed on its payload (if enabled), its header HEC value is calculated and the coset function is performed on the HEC (if enabled). The cell is then sent to the PHY via the SI. Once the cell transmission is complete, the relevant TC cell counters are updated.

An underrun condition occurs when a cell is to be sent to maintain the bit rate, but the TC transmit FIFO is empty. An idle cell is sent instead. This condition generates a TCER[UR] interrupt (if not masked) if TCMODE[URE] is set.

When a TC cell counter overflows, an interrupt is set (if enabled).

A TC channel is responsible for providing the data rate dictated by the PMD device. This is done by operating FCC2 in one of two modes:

- External Rate—In this mode the data rate is determined by the external device.The CP is responsible for keeping the FCC FIFO full.This is done by inserting ATM cells or idle cells (if ATM cells are not available) into the FCC FIFO whenever it is not full.This operation ensures that the cell stream is the data rate required by the PMD. The idle cells are supplied by the CP for the FCC FIFO. In general, the TC would never have its transmit FIFO empty, and thus would not need to generate idle cells. However, if the CP is busy, and the TC is forced to generate an idle cell because its transmit FIFO is empty, an underrun condition occurs. The UR interrupt is sent to the host (if not masked) if TCMODE[URE] is set. See Figure 38-12. This mode loads the CP more than using the internal rate mode.

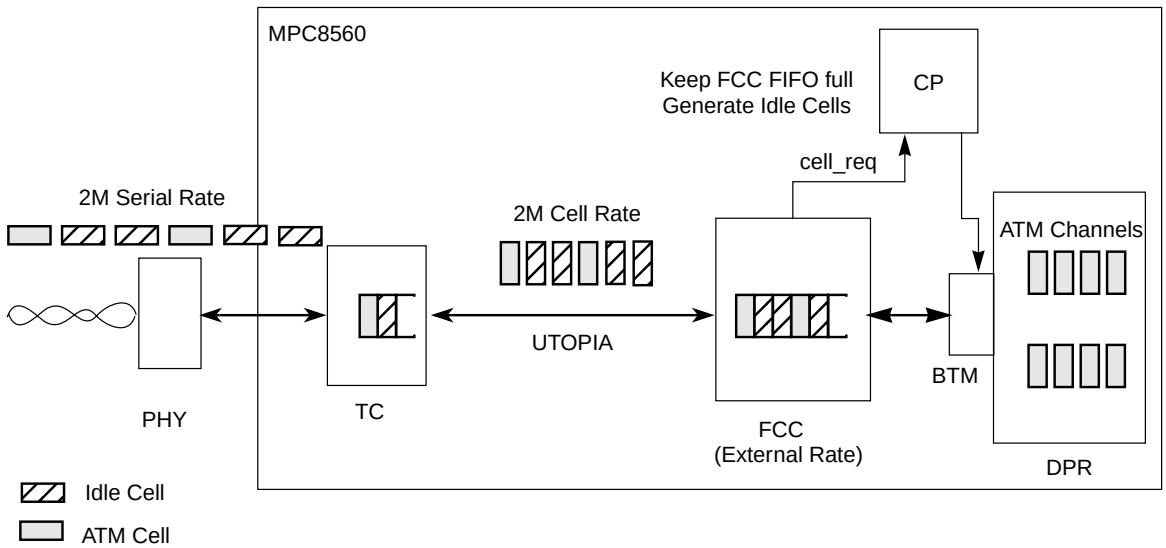


Figure 38-12. TC Operation in FCC External Rate Mode

- Internal Rate (Sub Rate)—In this mode only four PHY devices (which have four distinct addresses) are supported by the FCC. Each channel is controlled by a dedicated hardware timer that is programmed and tuned to the data rate needed. When a timer expires, a valid cell is sent to the corresponding PHY. The PHY sends idle cells to keep its synchronization and transmission rate. The same is true for the TC. Once activated with FCC2 in this mode, the TC keeps requesting cells and sends idle cells (UR interrupts can by disabled by clearing TCMODE[URE]) until a valid cell is transferred via the UTOPIA bus from the FCC. See Figure 38-13.

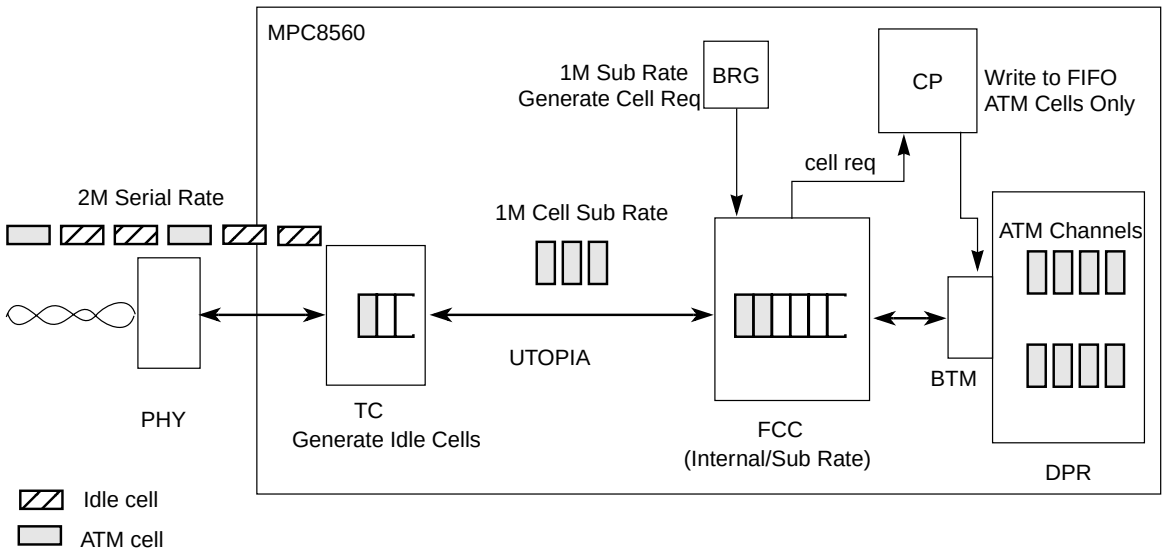


Figure 38-13. TC Operation in FCC Internal Rate Mode (Sub Rate Mode)

Operation in byte aligned mode (TCMODE[xTBA] = 1) is required for T1/E1 mainly. In this mode, once the TC is enabled, it waits for the first Txsyn pulse to start transmit the first byte of the first cell. This ensures that subsequent Txsyn pulses are byte-aligned to the cell boundaries.

38.12 TC Layer Implementation Example

Figure 38-14 shows the MPC8560 connected to two PHY devices, each containing four T1 framers. The eight T1 bit streams are connected to the eight MPC8560 SI TDMs and routed via the SI to the eight TC layer blocks. The eight TC layer blocks, each with its own address, are connected internally to FCC2 via the UTOPIA 8-bit bus. Another ATM stream is managed by FCC1 via the UTOPIA 16-bit bus connected to a SONET 155-Mbps PHY.

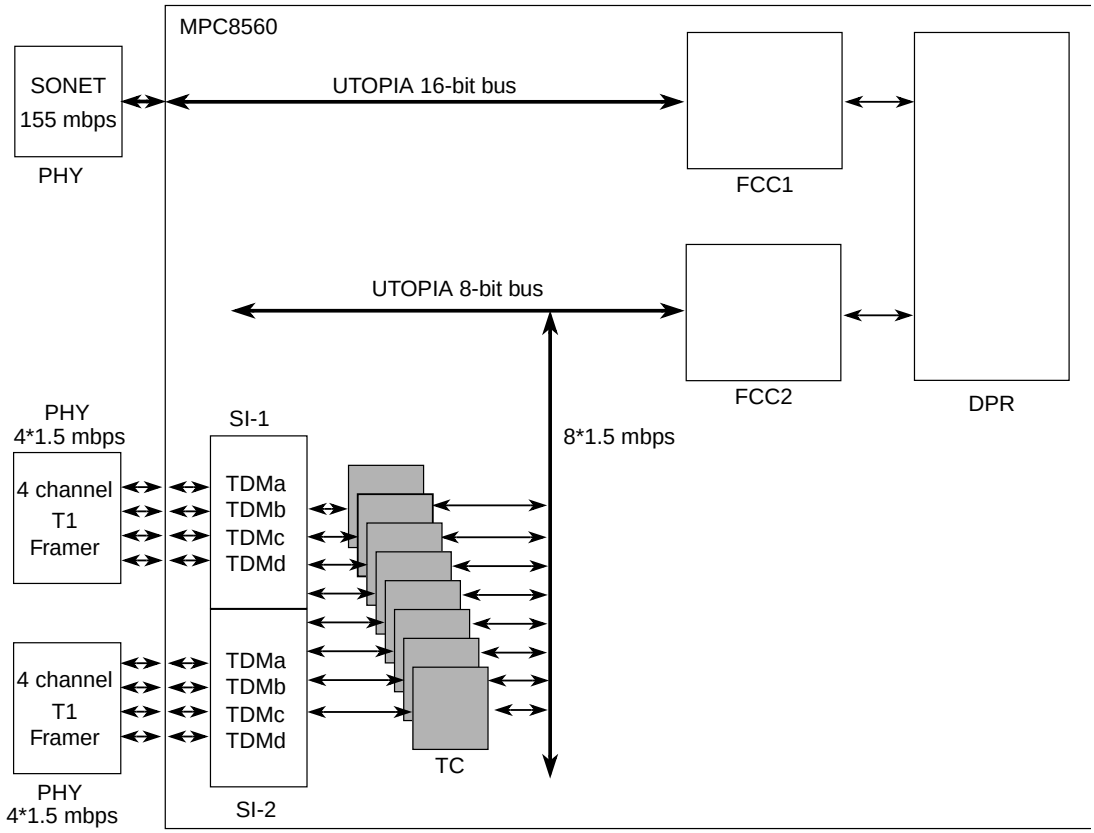


Figure 38-14. Example of Serial ATM Application

38.13 Operating the TC Layer at Higher Frequencies

The operation of the TC layer requires a minimum frequency ratio of 1:2.5 between the serial clock and the UTOPIA clock (in Rx and Tx separately). Using the TC for serial frequencies greater than 10 MHz requires using higher UTOPIA frequencies to preserve that ratio.

38.14 Programming a T1 Application

This section describes the configurations necessary to implement a T1 application using a single TC layer block. Note that using two or more TCs requires FCC2 to work in MPHY mode. Assuming that the required ATM parameters and data structures have been setup and initialized, implementing a T1 application requires the following steps:

1. Program FCC2
2. Setup I/O Ports and Clocks
3. Enable Tx/Rx on FCC2

4. Program CPM MUX
5. Program the TC block
6. Program the Serial Interface (SI)
7. Enable TDM

Step 1

To setup and initialize FCC2, program the FPSMR and GFMR as shown in Table 38-9. This is for working with one TC block operating in a single PHY environment. The transmitter and receiver should not be enabled at this time. In this example, FCC2 does not discard idle cells.

Table 38-9. Programming GFMR and FPSMR to Setup the FCC2

Init Values	Description
FPSMR2 = 0x0080_0000	UTOPIA Rx and Tx in Master Mode, Idle cells are not discarded
GFMR2 = 0x0000_000A	ATM Protocol Mode, Receiver and Transmitter are disabled

Step 2

Since the FCC2 UTOPIA bus is connected internally to the TC UTOPIA bus, program the parallel ports and BRGs for the active TDM(s).

Step 3

To enable receiving and transmitting, FCC2 should be programmed as shown in Table 38-10.

Table 38-10. Enable FCC2

Init Values	Description
GFMR2 = 0x0000_003A	Enable Rx and Tx

Step 4

To define the connection of FCC2, the CPM MUX should be programmed as shown in Table 38-11.

Table 38-11. Programming the CPM MUX for a TI Application

Init Values	Description
CMXFCR = 0x0080_0000	FCC2 is connected to the TC Layer
CMXUAR = 0x0000	FCC2 as UTOPIA master

Step 5

The TCx layer block should be configured using the TCMODEx and CDSMR1 registers as shown in Table 38-12. Note that the TC layer must be enabled after both FCC2 and CPM MUX have been programmed.

Table 38-12. Programming the TC Layer Block

Init Values	Description
TCMODE1 = 0xC202	Enable TC Layer Rx and Tx, no error correction on header, the TC is the only PHY on UTOPIA
CDSMR1 = 0x3980	ALPHA = 7, DELTA = 6 (default values)

Step 6

Program the SI to retrieve the data bits (192 bits) out of the T1 frame (193 bits). The SI frame pattern is programmed in the SI RAM (Rx or Tx), as shown in Table 38-13.

Table 38-13. Programming the SI RAM (Rx or Tx) for a T1 Application

Init Values	Description
SI_RAM[00]=0x0000	1 bit is ignored.
SI_RAM[02]=0x015E	Route 8 bytes to FCC2.
SI_RAM[04]=0x015E	Route 8 bytes to FCC2.
SI_RAM[06]=0x015F	Route 8 bytes to FCC2 and go back to the first entry in table.

Step 7

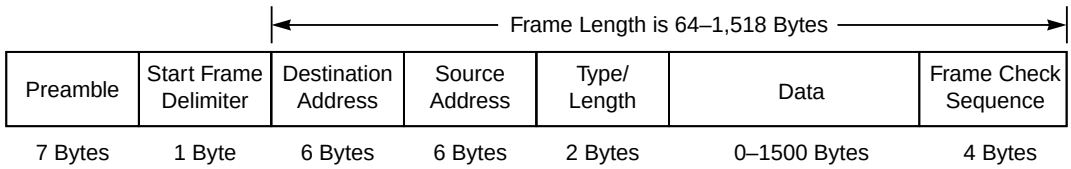
The last step in this example is to initialize the serial interface registers and enable TDMx—in this case TDMa on SI1 as shown in Table 38-14.

Table 38-14. Programming SI registers to enable TDM

Init Values	Description
SI1AMR = 0x0040	Common Receive and Transmit Pins for TDMa
SI1GMR = 01	Enable TDMa

Chapter 39
CPM Ethernet Controller

Ethernet is a local area network (LAN) protocol defined in the IEEE 802.3 standard in which computers access the network through a Carrier Sense Multiple Access / Collision Detect (CSMA/CD) protocol. 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. Ethernet/IEEE 802.3 frames have the structure shown in Figure 39-1.



Note: The lsb of each octet is transmitted first.

Figure 39-1. Ethernet Frame Structure

The elements of an Ethernet frame are as follows:

- 7-byte preamble of alternating ones and zeros (used to synchronize the PLL detection electronics)
- Start frame delimiter (SFD)—Signifies the beginning of the frame.
- 48-bit destination address.
- 48-bit source address. 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 requirement limits the length of the data portion of the frame to 1,500 bytes and, therefore, the total frame length to 1,518 bytes.
- Data—If the size of this field is less than 46 bytes, then use of the “Pad” field is necessary to bring the frame size up to the minimum length.

- Four-byte frame-check sequence (FCS), which is the standard, 32-bit CCITT-CRC polynomial used in many protocols.

39.1 Ethernet Protocol Basics

All Ethernet stations monitor the channel (receive) at all times. When a station has data to transmit, it waits until the channel has been silent for a specified time period (interframe gap), and then simply begins transmitting. Typically the frame transmission completes normally – provided no other station begins transmitting at the same time.

Two stations “collide” when they attempt to claim the channel simultaneously. Because all transmitting stations are also receiving – even when they’re transmitting – they can detect collisions by simply comparing transmitted and received data streams. A collision event occurs whenever these streams don’t match, which is usually detected near the beginning of a frame. When a station detects a collision, it executes the following three steps.

1. jams the channel (transmits all ones) to ensure the other transmitting station(s) also detect the collision
2. stops transmitting
3. waits a random time period (backoff delay) before attempting retransmission

After the backoff delay, colliding stations again wait for silence on the channel and then begin transmitting. This process of jamming the channel, waiting, and retransmitting, is called a retry. If the frame is not successfully sent within 15 retries, the MAC function reports an “excessive collision error”. The frame being transmitted is then dropped, requiring the application software to detect its loss and initiate a retransmission.

Some basic timing specifications for 10-Mbps and 100-Mbps Ethernet are shown in Table 39-1. Additional information about Ethernet, including specifications, definitions, and development history can be found on the IEEE web site or at the following URL:
http://www.techfest.com/networking/lan/ethernet.htm

Table 39-1. Basic Ethernet Timing Specifications

Parameter	Size in bits	Size in µs	
		10-Mbps	100-Mbps
byte length	8	0.8	0.08
preamble plus start frame delimiter length	64	6.4	0.64
minimum interframe gap	96	9.6	0.96
slot time (minimum frame transmission length)	512	51.2	5.12

39.2 Fast Ethernet on the MPC8560 CPM

When a general FCC mode register (GFMR*x*[MODE]) selects Ethernet protocol, that FCC performs the full set of IEEE 802.3/Ethernet CSMA/CD media access control (MAC) and channel interface functions. Figure 39-2 shows a block diagram of the FCC Ethernet control logic.

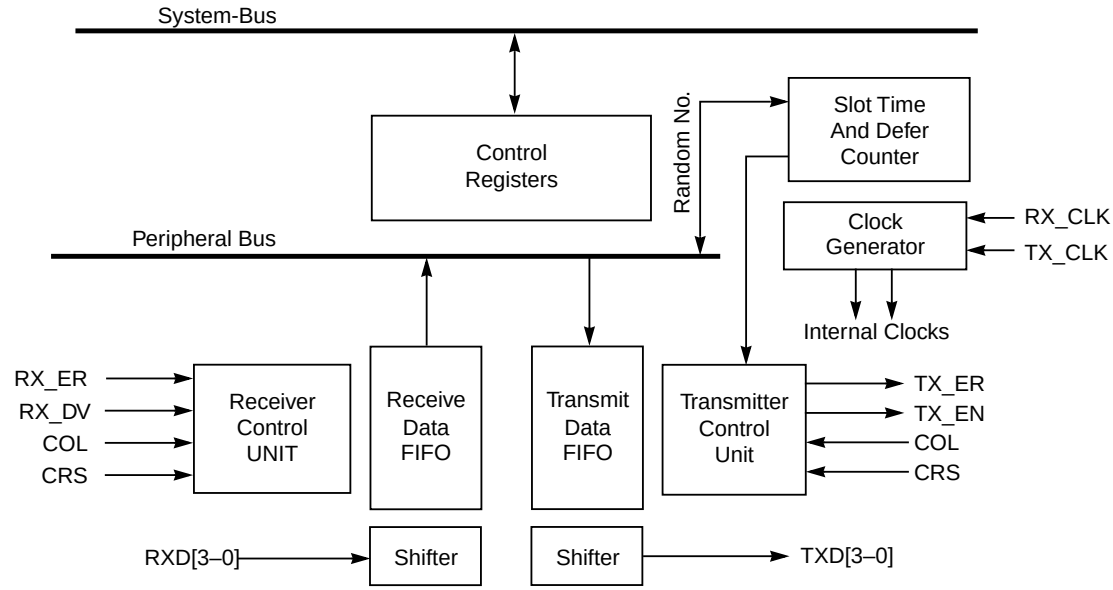


Figure 39-2. Ethernet Block Diagram

39.3 Features

The following is a list of CPM fast Ethernet key features:

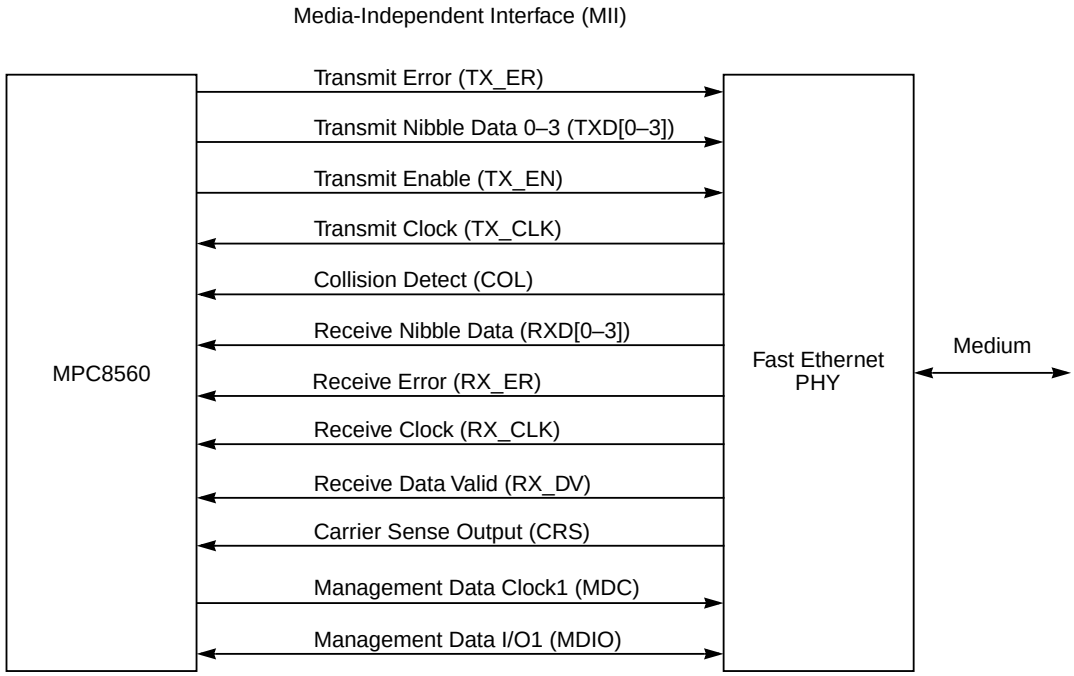
- Support for fast Ethernet through the MII (media-independent interface)
- Performs MAC (media access control) layer functions of fast Ethernet and IEEE 802.3x
- Performs framing functions
  - — Preamble generation and stripping
  - — Destination address checking
  - — CRC generation and checking
  - — Automatic padding of short frames on transmit
  - — Framing error (dribbling bits) handling
- Full collision support
  - — Enforces the collision (jamming and TX_ER assertion)
  - — Truncated binary exponential backoff algorithm for random wait

  - Two nonaggressive backoff modes
  - Automatic frame retransmission (until retry limit is reached)
  - Automatic discard of incoming collided frames
  - Delay transmission of new frames for specified interframe gap
- Bit rates up to 100 Mbps
- Receives back-to-back frames
- Detection of receive frames that are too long
- Multibuffer data structure
- Supports 48-bit addresses in three modes
  - Physical. One 48-bit address recognized or 64-bin hash table for physical addresses
  - Logical. 64-bin group address hash table plus broadcast address checking
  - Promiscuous. Receives all frames regardless of address (a CAM can be used for address filtering)
- External CAM support on system bus interfaces
- Special RMON counters for monitoring network statistics
- Transmitter network management and diagnostics
  - Lost carrier sense
  - Underrun
  - Number of collisions exceeded the maximum allowed
  - Number of retries per frame
  - Deferred frame indication
  - Late collision
- Receiver network management and diagnostics
  - CRC error indication
  - Nonoctet alignment error
  - Frame too short
  - Frame too long
  - Overrun
  - Busy (out of buffers)
- Error counters
  - Discarded frames (out of buffers or overrun occurred)
  - CRC errors
  - Alignment errors

- Internal and external loopback mode
- Supports fast Ethernet in duplex mode
- Supports pause flow control frames
- Support of out-of-sequence transmit queue (for flow-control frames)
- External buffer descriptors (BDs)

39.4 Connecting the MPC8560 to Fast Ethernet

Figure 39-3 shows the basic components of the media-independent interface (MII) and the signals required to make the fast Ethernet connection between the MPC8560 and a PHY.



[1] The management signals (MDC and MDIO) can be common to all of the fast Ethernet connections in the system, assuming that each PHY has a different management address. Use parallel I/O port pins to implement MDC and MDIO. (The I^2C controller cannot be used for this function.)

Figure 39-3. Connecting the MPC8560 to Ethernet

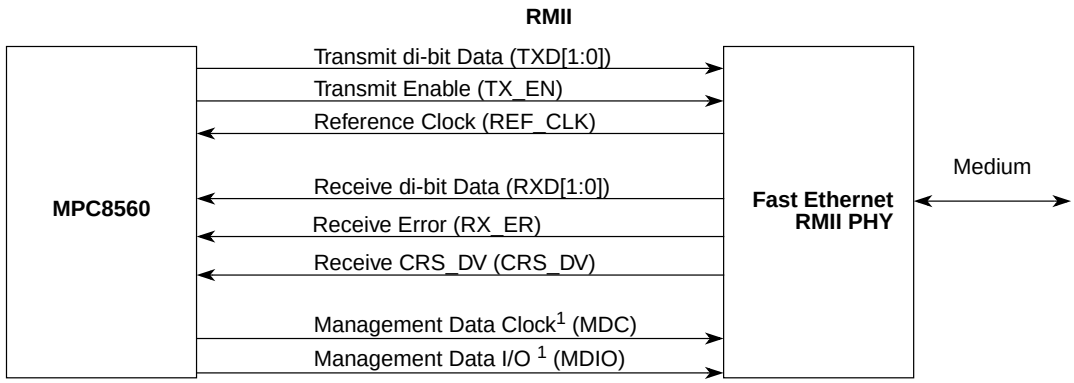
Each FCC has 18 signals, defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY. The two management signals (MDC and MDIO) required by the MII should be implemented separately using the parallel I/O.

The MPC8560 has additional signals for interfacing with an optional external content-addressable memory (CAM), which are described in Section 39.8, “CAM Interface.”

The MPC8560 uses the SDMA channels to store every byte received after the start frame delimiter into system memory. On transmit, the user provides the destination address, source address, type/length field, and transmit data. To meet minimum frame requirements, MPC8560 automatically pads frames with fewer than 64 bytes in the data field. The MPC8560 also appends the FCS to the frame.

39.4.1 Connecting the MPC8560 to Ethernet (RMII)

Figure 39-4 shows the basic components of the reduced media-independent interface (RMII) and the signals required for the fast Ethernet connection between the MPC8560 and a PHY. The MDC/MDIO management interface is the same as in MII. The RMII reference clock (REF_CLK) is distributed over the FCC transmit clock. In RMII mode receive clock is not used.



[1]The management signals (MDC and MDIO) can be common to all of the fast Ethernet connections in the system, assuming that each PHY has a different management address. Use parallel I/O port pins to implement MDC and MDIO. (The I^2C controller cannot be used for this function.)

Figure 39-4. Connecting the MPC8560 to Ethernet (RMII)

39.5 Ethernet Channel Frame Transmission

The Ethernet transmitter requires almost no core intervention. When the core enables the transmitter, the Ethernet controller polls the first TxBD in the FCC's TxBD table every 256 serial clocks. If the user has a frame ready to transmit, setting FTODR[TOD] eliminates waiting for the next poll. When there is a frame to transmit, the Ethernet controller begins fetching the data from the data buffer and asserts TX_EN. The preamble sequence, start frame delimiter, and frame information are sent in that order; see Figure 39-1 on page 39-1. In full-duplex mode, because collisions are ignored, frame transmission maintains only the interframe gap 27 serial clocks (108-bit time period) regardless of CRS assertion.

There is one internal buffer for out-of-sequence flow control frames (in full-duplex fast Ethernet). When the fast Ethernet controller is between frames, this buffer is polled if flow control is enabled. This buffer must contain the whole frame.

However, in half-duplex mode, the controller defers transmission if the line is busy (CRS asserted). Before transmitting, the controller waits for carrier sense to become inactive, at which point the controller determines if CRS remains negated for 60 serial clocks. If so, the transmission begins after an additional 36 serial clocks (96 serial clocks after CRS originally became negated). In the fast ethernet transmitter, if CRS is asserted and then negated within 10 clocks after TXEN is negated, the next frame is not deferred and a defer indication is asserted.

If a collision occurs during the transmit frame, the Ethernet controller follows a specified backoff procedure and tries to retransmit the frame until the retry limit is reached. The Ethernet controller stores at least the first 64 bytes of data of the transmit frame in the FCC FIFO, so that the data does not have to be retrieved from system memory in case of a collision. This improves bus usage and latency if the backoff timer output requires an immediate retransmission.

When the end of the current buffer is reached and TxBD[L] = 1, the FCS (32-bit CRC) bytes are appended (if TxBD[TC] = 1), and TX_EN is negated. This notifies the PHY of the need to generate the illegal Manchester encoding that signifies the end of an Ethernet frame. Following the transmission of the FCS, the Ethernet controller writes the frame status bits into the BD and clears TxBD[R]. When the end of the current buffer is reached and TxBD[L] = 0 (a frame is comprised of multiple buffers), 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] = 1, the Ethernet controller pads short frames to the value of the minimum frame length register (MINFLR), described in Table 39-3 on page 39-10.

To rearrange the transmit queue before the CP finishes sending all frames, issue a GRACEFUL STOP TRANSMIT command. This can be useful for transmitting expedited data ahead of previously linked buffers or for error situations. When the GRACEFUL STOP TRANSMIT command is issued, the Ethernet controller stops immediately if no transmission is in progress or continues transmission until the current frame either finishes or terminates with a collision. When the Ethernet controller is given the RESTART TRANSMIT command, it resumes transmission. The Ethernet controller sends bytes least-significant nibble first.

39.6 Ethernet Channel Frame Reception

The Ethernet receiver is designed to work with almost no core intervention and can perform address recognition, CRC checking, short frame checking, maximum DMA transfer checking, and maximum frame-length checking.

When the core enables the Ethernet receiver, it enters hunt mode when RX_DV is asserted as long as COL remains negated (full-duplex mode ignores COL). In hunt mode, as data is shifted into the receive shift register four bits at a time, the contents of the register are

compared to the contents of the SYN2 field in the FCC's data synchronization register (FDSR). When the registers match, the hunt mode is terminated and character assembly begins.

When the receiver detects the first bytes of a frame, the Ethernet controller performs address recognition functions on the frame; see Section 39.13, "Ethernet Address Recognition." The receiver can receive physical (individual), group (multicast), and broadcast addresses. Because Ethernet receive frame data is not written to memory until the internal address recognition algorithm is complete, bus usage is not wasted on frames not addressed to this station. The receiver can also operate with an external CAM, in which case frame reception continues normally, unless the CAM specifically signals the frame to be rejected. See Section 39.8, "CAM Interface."

If an address is recognized, the Ethernet controller fetches the next RxBD and, if it is empty, starts transferring the incoming frame to the RxBD's associated data buffer.

In half-duplex mode, if a collision is detected during the frame, the RxBDs associated with this frame are reused. Thus, no collision frames are presented to the user except late collisions, which indicate serious LAN problems. When the buffer has been filled, the Ethernet 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.

The RxBD length is determined by MRBLR in the parameter RAM. The user should program MRBLR to be at least 64 bytes. During reception, the Ethernet controller checks for frames that are too short or too long. When the frame ends, the receive CRC field is checked and written to the data buffer. The data length written to the last BD in the Ethernet frame is the length of the entire frame, which enables the software to recognize a frame-too-long condition.

If an external CAM is used (FPSMR*x*[CAM] = 1), the Ethernet controller adds the two lower bytes of the CAM output at the end of each frame. Note that the data length does not include these two bytes; that is, the extra two bytes could push the buffer length past MRBLR.

When the receive frame is complete, the Ethernet controller sets RxBD[L], writes the other frame status bits into the RxBD, and clears RxBD[E]. 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. The Ethernet controller receives serial data least-significant nibble first.

39.7 Flow Control

Because collisions cannot occur in full-duplex mode, fast 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 39-2 shows the flow-control frame structure.

Table 39-2. 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	up to 0xFFFE	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: MAC Parameter ValuePause Period 0 none 1 or 22 x slot time 3 or 44 x slot time ……
42	Reserved	—	
4	FCS		Frame check sequence (CRC)

When flow-control mode is enabled (FPSMR*x*[FCE]) and the receiver identifies a pause-flow control frame sent to individual or broadcast addresses, transmission stops for the time specified in the control frame. During this pause, only the out-of-sequence frame is sent. 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.

39.8 CAM Interface

The MPC8560 internal address recognition logic can be used in combination with an external CAM. When using a CAM, the FCC must be in promiscuous mode (FPSMR*x*[PRO] = 1). See Section 39.13, “Ethernet Address Recognition.”

The Ethernet controller writes two 32-bit accesses to the CAM and then reads the result in a 32-bit access. If the high bit of the result is set, the frame is rejected; otherwise, the lower 16 bits are attached to the end of the frame.

When an external CAM is used for address filtering, users can choose to either discard rejected frames (FPSMR[ECM] = 0) or receive rejected frames and signal the CAM miss in the RxBD (FPSMR[ECM] = 1).

39.9 Ethernet Parameter RAM

For Ethernet mode, the protocol-specific area of the FCC parameter RAM is mapped as in Table 39-3.

Table 39-3. Ethernet-Specific Parameter RAM

Offset [1]	Name	Width	Description
0x3C	STAT_BUF	Word	Buffer of internal usage
0x40	CAM_PTR	Word	CAM address
0x44	C_MASK	Word	Constant MASK for CRC (initialize to 0xDEBB_20E3). For the 32-bit CRC-CCITT.
0x48	C_PRES	Word	Preset CRC (initialize to 0xFFFF_FFFF). For the 32-bit CRC-CCITT.
0x4C	CRCEC [2]	Word	CRC error counter. Counts each received frame with a CRC error. Does not count frames not addressed to the station, frames received in the out-of-buffers condition, frames with overrun errors, or frames with alignment errors.
0x50	ALEC [2]	Word	Alignment error counter. Counts frames received with dribbling bits. Does not count frames not addressed to the station, frames received in the out-of-buffers condition, or frames with overrun errors.
0x54	DISFC [2]	Word	Discard frame counter. Incremented for discarded frames because of an out-of-buffers condition or overrun error. The CRC need not be correct for this counter to be incremented.
0x58	RET_LIM	Hword	Retry limit (typically 15 decimal). Number of retries that should be made to send a frame. If the frame is not sent after this limit is reached, an interrupt can be generated.
0x5A	RET_CNT	Hword	Retry limit counter. Temporary decrementer used to count retries made.
0x5C	P_PER	Hword	Persistence. Allows the Ethernet controller to be less persistent after a collision. Normally cleared, P_PER can be from 0 to 9 (9 = least persistent). The value is added to the retry count in the backoff algorithm to reduce the chance of transmission on the next time-slot. Using a less persistent backoff algorithm increases throughput in a congested Ethernet LAN by reducing the chance of collisions. FPSMR[SBT] can also reduce persistence of the Ethernet controller. The Ethernet/802.3 specifications permit the use of P_PER.
0x5E	BOFF_CNT	Hword	Backoff counter
0x60	GADDR_H	Word	Group address filters high and low are used in the hash table function of the group addressing mode. The user may write zeros to these values after reset and before the Ethernet channel is enabled to disable all group hash address recognition functions. The SET GROUP ADDRESS command is used to enable the hash table. See Section 39.14, “Hash Table Algorithm.”
0x64	GADDR_L	Word	
0x68	TFCSTAT	Hword	Out-of-sequence TxBD. Includes the status/control, data length, and buffer pointer fields in the same format as a regular TxBD. Useful for sending flow control frames. This area’s TxBD[R] is always checked between frames, regardless of FPSMR*x*[FCE]. If it is not ready, a regular frame is sent. The user must set TxBD[L] when preparing this BD. If TxBD[I] is set, a TXC event is generated after frame transmission. This area should be cleared when not in use.
0x6A	TFCLEN	Hword	
0x6C	TFCPTR	Word	

Table 39-3. Ethernet-Specific Parameter RAM (continued)

Offset [1]	Name	Width	Description
0x70	MFLR	Hword	Maximum frame length register (typically1518 decimal). If the Ethernet controller detects an incoming frame exceeding MFLR, it sets RxBD[LG] (frame too long) in the last RxBD, but does not discard the rest of the frame. The controller also reports the frame status and length of the received frame in the last RxBD. MFLR includes all in-frame bytes between the start frame delimiter and the end of the frame.
0x72	PADDR1_H	Hword	The 48-bit individual address of this station. PADDR1_L is the lowest order half-word, and PADDR1_H is the highest order half-word.
0x74	PADDR1_M	Hword	
0x76	PADDR1_L	Hword	
0x78	IBD_CNT	Hword	Internal BD counter
0x7A	IBD_START	Hword	Internal BD start pointer
0x7C	IBD_END	Hword	Internal BD end pointer
0x7E	TX_LEN	Hword	Tx frame length counter
0x80	IBD_BASE	32 Bytes	Internal microcode usage
0xA0	IADDR_H	Word	Individual address filter high/low. Used in the hash table function of the individual addressing mode. The user can write zeros to these values after reset and before the Ethernet channel is enabled to disable all individual hash address recognition functions. Issuing a SET GROUP ADDRESS command enables the hash table. See Section 39.14, “Hash Table Algorithm.”
0xA4	IADDR_L	Word	
0xA8	MINFLR	Hword	Minimum frame length register (typically 64 decimal). If the Ethernet receiver detects an incoming frame shorter than MINFLR, it discards that frame unless FPSMR[RSH] (receive short frames) is set, in which case RxBD[SH] (frame too short) is set in the last RxBD. The Ethernet transmitter pads frames that are too short (according to TxBD[PAD] and the PAD value in the parameter RAM). PADs are added to make the transmit frame MINFLR bytes.
0xAA	TADDR_H	Hword	Allows addition of addresses to the individual and group hashing tables. After an address is placed in TADDR, issue a SET GROUP ADDRESS command. TADDR_L is the lowest-order half-word; TADDR_H is the highest. A zero in the I/G bit indicates an individual address; 1 indicates a group address.
0xAC	TADDR_M	Hword	
0xAE	TADDR_L	Hword	
0xB0	PAD_PTR	Hword	Internal PAD pointer. This internal 32-byte aligned pointer points to a 32-byte buffer filled with pad characters. The pads may be any value, but all the bytes should be the same to assure padding with a specific character. If a specific padding character is not needed, PAD_PTR should equal the internal temporary data pointer TIPTR; see Section 34.8, “FCC Parameter RAM.”
0xB2	—	Hword	Reserved, should be cleared.
0xB4	CF_RANGE	Hword	Control frame range. Internal usage
0xB6	MAX_B	Hword	Maximum BD byte count. Internal usage
0xB8	MAXD1	Hword	Max DMA1 length register (typically 1520 decimal). Lets the user prevent system bus writes after a frame exceeds a specified size. The MAXD1 value is valid only if an address match is detected. If the Ethernet controller detects an incoming Ethernet frame larger than the user-defined value in MAXD1, the rest of the frame is discarded. The Ethernet controller waits for the end of the frame (or until MFLR bytes have been received) and reports the frame status and length (including the discarded bytes) in the last RxBD. This value must be greater than 32.

Table 39-3. Ethernet-Specific Parameter RAM (continued)

Offset [1]	Name	Width	Description
0xBA	MAXD2	Hword	Max DMA2 length register (typically 1520 decimal). Lets the user prevent system bus writes after a frame exceeds a specified size. The value of MAXD2 is valid in promiscuous mode when no address match is detected. If the Ethernet controller detects an incoming Ethernet frame larger than the value in MAXD2, the rest of the frame is discarded. The Ethernet controller waits for the end of the frame (or until MFLR bytes are received) and reports frame status and length (including the discarded bytes) in the last RxBD. In a monitor station, MAXD2 can be much less than MAXD1 to receive entire frames addressed to this station, but receive only the headers of all other frames.This value must be less than MAXD1.
0xBC	MAXD	Hword	Rx maximum DMA. Internal usage
0xBE	DMA_CNT	Hword	Rx DMA counter. Temporary down-counter used to track the frame length.
0xC0	OCTC [2]	Word	(RMON mode only) The total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including FCS octets).
0xC4	COLC [2]	Word	(RMON mode only) The best estimate of the total number of collisions on this Ethernet segment.
0xC8	BROC [2]	Word	(RMON mode only) The total number of good packets received that were directed to the broadcast address. Note that this does not include multicast packets.
0xCC	MULC [2]	Word	(RMON mode only) The total number of good packets received that were directed to a multicast address. Note that this number does not include packets directed to the broadcast address.
0xD0	USPC [2]	Word	(RMON mode only) The total number of packets received that were less than 64 octets (excluding framing bits but including FCS octets) and were otherwise well-formed.
0xD4	FRGC [2]	Word	(RMON mode only) The total number of packets received that were less than 64 octets long (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Note that it is entirely normal for etherStatsFragments to increment because it counts both runts (which are normal occurrences due to collisions) and noise hits.
0xD8	OSPC [2]	Word	(RMON mode only) The total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets) and were otherwise well-formed.
0xDC	JBRC [2]	Word	(RMON mode only) The total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets), and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Note that this definition of jabber is different than the definition in IEEE-802.3 section 8.2.1.5 (10BASE5) and section 10.3.1.4 (10BASE2). These documents define jabber as the condition where any packet exceeds 20 ms. The allowed range to detect jabber is between 20 ms and 150 ms.
0xE0	P64C [2]	Word	(RMON mode only) The total number of packets (including bad packets) received that were 64 octets long (excluding framing bits but including FCS octets).
0xE4	P65C [2]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 65 and 127 octets long inclusive (excluding framing bits but including FCS octets).

Table 39-3. Ethernet-Specific Parameter RAM (continued)

Offset [1]	Name	Width	Description
0xE8	P128C [2]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 128 and 255 octets long inclusive (excluding framing bits but including FCS octets).
0xEC	P256C [2]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 256 and 511 octets long inclusive (excluding framing bits but including FCS octets).
0xF0	P512C [2]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 512 and 1023 octets long inclusive (excluding framing bits but including FCS octets).
0xF4	P1024C [2]	Word	(RMON mode only) The total number of packets (including bad packets) received that were between 1024 and 1518 octets long inclusive (excluding framing bits but including FCS octets).
0xF8	CAM_BUF	Word	Internal buffer for CAM result
0xFC	—	Word	Reserved, should be cleared.

1 Offset from FCC base: 0x8400 (FCC1), 0x8500 (FCC2) and 0x8600 (FCC3); see Section 20.5.2, “Parameter RAM.”
2 32-bit (modulo 232) counters maintained by the CP; cleared by the user while the channel is disabled.

39.10 Programming Model

The core configures an FCC to operate as an Ethernet controller using GFMR[MODE]. The receive errors (collision, overrun, nonoctet-aligned frame, short frame, frame too long, and CRC error) are reported through the RxBD. The transmit errors (underrun, heartbeat, late collision, retransmission limit, and carrier sense lost) are reported through the TxBD.

The user should program the FDSR as described in Section 34.5, “FCC Data Synchronization Registers (FDSRx),” with FDSR[SYN2] = 0xD5 and FDSR[SYN1] = 0x55.

39.11 Ethernet Command Set

The transmit and receive commands are issued to the CPCR; see Section 20.4, “Command Set.”

NOTE

Before resetting the CPM, configure TX_EN ($\overline{\text{RTS}}$) as an input.

Transmit commands that apply to Ethernet are described in Table 39-4.

Table 39-4. Transmit Commands

Command	Description
STOP TRANSMIT	When used with the Ethernet controller, this command violates a specific behavior of an Ethernet/IEEE 802.3 station. It should not be used.
GRACEFUL STOP TRANSMIT	Used to smoothly stop transmission after the current frame finishes sending or undergoes a collision (immediately if there is no frame being sent). FCCE[GRA] is set once transmission stops. Then the Ethernet transmit parameters (including BDs) can be modified by the user. The TBPTR points to the next TxBD in the table. Transmission begins when the R bit of the next BD is set and the RESTART TRANSMIT command is issued. Note that if the GRACEFUL STOP TRANSMIT command is issued and the current transmit frame ends in a collision, the TBPTR points to the beginning of the collided frame with TxBD[R] still set (the frame looks as if it was never sent).
RESTART TRANSMIT	Enables transmission of characters on the transmit channel. It is expected by the Ethernet controller after a GRACEFUL STOP TRANSMIT command or transmitter error (underrun, retransmission limit reached, or late collision). The Ethernet controller resumes transmission from the current TBPTR in the channel TxBD table.
INIT TX PARAMETERS	Initializes all the transmit parameters in this serial channel parameter RAM to their reset state. This command should be issued only when the transmitter is disabled. Note that the INIT TX AND RX PARAMETERS command can also be used to reset the transmit and receive parameters.

Receive commands that apply to Ethernet are described in Table 39-5.

Table 39-5. Receive Commands

Command	Description
ENTER HUNT MODE	After the hardware or software is reset and the channel in the FCC mode register is enabled, the channel is in the receive enable mode and uses the first BD in the table. This command is generally used to force the Ethernet receiver to abort reception of the current frame and enter hunt mode. In hunt mode, the Ethernet controller continually scans the input data stream for a transition of carrier sense from inactive to active followed by a preamble sequence and the start frame delimiter. After receiving the command, the current receive buffer is closed, the error status flags and length field are cleared, RxBD[E] (the empty bit) is set, and the CRC calculation is reset. Further frame reception uses the current RxBD. Note that short frames pending in the internal FIFO may be lost.
INIT RX PARAMETERS	Initializes all the receive parameters in this serial channel parameter RAM to their reset state and should only be issued when the receiver is disabled. Note that the INIT TX AND RX PARAMETERS command can also be used to reset the receive and transmit parameters.
SET GROUP ADDRESS	Used to set one of the 64 bits of the four individual/group address hash filter registers (GADDR[1–4] or IADDR[1–4]). The individual or group address (48 bits) to be added to the hash table should be written to TADDR_L, TADDR_M, and TADDR_H in the parameter RAM prior to executing this command. The CP checks the I/G bit in the address stored in TADDR to determine whether to use the individual hash table or the group hash table. A 0 in the I/G bit indicates an individual address; 1 indicates a group address. This command can be executed at any time, regardless of whether the Ethernet channel is enabled.

If an address from the hash table must be deleted, the Ethernet channel must be disabled, the hash table registers must be cleared, and the SET GROUP ADDRESS command must be executed for the remaining preferred addresses. This is required because the hash table might have mapped multiple addresses to the same hash table bit.

39.12 RMON Support

The fast Ethernet controller can automatically gather network statistics required for RMON without the need to receive all addresses using promiscuous mode. Setting FPSMR*x*[MON] enables RMON support.

The RMON statistics and their corresponding counters in the parameter RAM are described in Table 39-6.

Table 39-6. RMON Statistics and Counters

Statistic	Description	Counter
etherStatsDropEvents	The total number of events in which packets were detected as dropped by the probe due to lack of resources. Note that this may not be the number of packets dropped; it is the number of times this condition is detected.	DISFC
etherStatsOctets	The total number of octets of data (including those in bad packets) received on the network (excluding framing bits but including FCS octets).	OCTC
etherStatsPkts	The total number of packets (including bad packets, broadcast packets, and multicast packets) received.	USPC + OSPC + FRGC + JBRC + P64C + P65C + P128C + P256C + P512C + P1024C
etherStatsBroadcastPkts	The total number of good packets received that were directed to the broadcast address. Note that this does not include multicast packets.	BROC
etherStatsMulticastPkts	The total number of good packets received that were directed to a multicast address. Note that this number does not include packets directed to the broadcast address.	MULC
etherStatsCRCAlignErrors	The total number of packets received that had a length (excluding framing bits but including FCS octets) of between 64 and 1518 octets, inclusive, but had either an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error).	CRCEC + ALEC - FRGC - JBRC
etherStatsUndersizePkts	The total number of packets received that were less than 64 octets long (excluding framing bits but including FCS octets) and were otherwise well-formed.	USPC
etherStatsOversizePkts	The total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets) and were otherwise well-formed.	OSPC

Table 39-6. RMON Statistics and Counters (continued)

Statistic	Description	Counter
etherStatsFragments	The total number of packets received that were less than 64 octets long (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Note that it is entirely normal for etherStatsFragments to increment, because it counts both runts (which are normal occurrences due to collisions) and noise hits.	FRGC
etherStatsJabbers	The total number of packets received that were longer than 1518 octets (excluding framing bits but including FCS octets) and had either a bad FCS with an integral number of octets (FCS error) or a bad FCS with a non-integral number of octets (alignment error). Note that this definition of jabber is different than the definition in IEEE-802.3 Section 8.2.1.5 (10BASE5) and Section 10.3.1.4 (10BASE2). These documents define jabber as the condition where any packet exceeds 20 ms. The allowed range to detect jabber is between 20 ms and 150 ms.	JBRC
etherStatsCollisions	The best estimate of the total number of collisions on this Ethernet segment.	COLC
etherStatsPkts64Octets	The total number of packets (including bad packets) received that were 64 octets long (excluding framing bits but including FCS octets).	P64C
etherStatsPkts65to127Octets	The total number of packets (including bad packets) received that were between 65 and 127 octets long inclusive (excluding framing bits but including FCS octets).	P65C
etherStatsPkts128to255Octets	The total number of packets (including bad packets) received that were between 128 and 255 octets long inclusive (excluding framing bits but including FCS octets).	P128C
etherStatsPkts256to511Octets	The total number of packets (including bad packets) received that were between 256 and 511 octets long inclusive (excluding framing bits but including FCS octets).	P256C
etherStatsPkts512to1023Octets	The total number of packets (including bad packets) received that were between 512 and 1023 octets long inclusive (excluding framing bits but including FCS octets).	P512C
etherStatsPkts1024to1518Octets	The total number of packets (including bad packets) received that were between 1024 and 1518 octets long inclusive (excluding framing bits but including FCS octets).	P1024C

39.13 Ethernet Address Recognition

The Ethernet controller can filter the received frames based on different addressing types—physical (individual), group (multicast), broadcast (all-ones group address), and promiscuous. The difference between an individual address and a group address is determined by the I/G bit in the destination address field.

Figure 39-5 is a flowchart for address recognition on received frames.

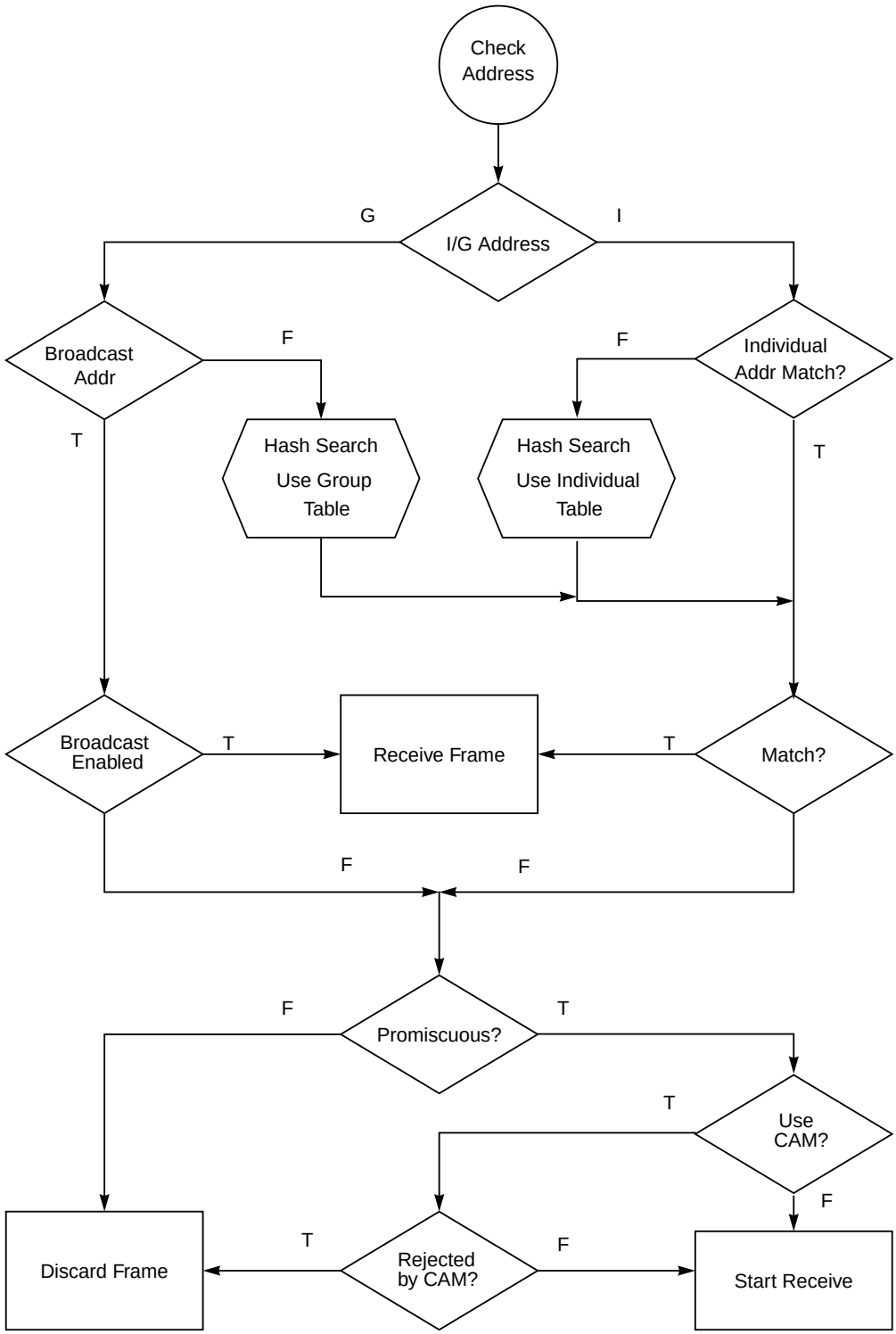


Figure 39-5. Ethernet Address Recognition Flowchart

In the physical type of address recognition, the Ethernet controller compares the destination address field of the received frame with the physical address that the user programs in the PADDR. If it fails, the controller performs address recognition on multiple individual addresses using the IADDR_H/L hash table. Since the controller always checks PADDR and the individual hash, for individual address the user must write zeros to the hash in order to avoid a hash match and ones to PADDR in order to avoid individual address match.

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_H/L hash table. In promiscuous mode, the Ethernet controller receives all of the incoming frames regardless of their address when an external CAM is not used.

If an external CAM is used for address recognition (FPSMR[CAM] = 1), the user should select promiscuous mode; the frame can be rejected if there is no match in the CAM. If the on-chip address recognition functions detect a match, the external CAM is not accessed. Otherwise, the CPM issues a match transaction to the CAM using the bus on which the data buffers reside. (The data buffer bus is selected in FCR*x*[DTB]; see Section 34.8.1, “FCC Function Code Registers (FCRx).”) Note that even if the CAM is placed on the local bus and the data buffers are on the system bus, match transactions are eventually accessed correctly. That is, the CPM attempts to access the CAM on the system bus, but the system-to-local-bus bridge logic detects the system bus transaction and forwards it to the CAM on the local bus.

39.14 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 address into one of 64 bins, which are represented by the 64 bits in GADDR_H/L or IADDR_H/L. When the SET GROUP ADDRESS command is executed, the Ethernet controller maps the selected 48-bit address in TADDR into one of the 64 bits. This is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and using 6 bits of the CRC-encoded result to generate a number between 1 and 64. Bit 26 of the CRC result selects between the two GADDRs or IADDRs; bits 27–31 of the CRC result select which bit is set.

The same process is used when the Ethernet controller receives a frame. If the CRC generator selects a bit that is set in the group/individual hash table, the frame is accepted; otherwise, it is rejected. The result is that if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (87.5%) of the group address frames from reaching memory. The core must further filter those that reach memory to determine if they contain one of the eight preferred addresses.

Better performance is achieved by using the group and individual hash tables in combination. For instance, if eight group and eight physical addresses are stored in their

respective hash tables, 87.5% of all frames (not just group address frames) are prevented from reaching memory.

The effectiveness of the hash table declines as the number of addresses increases. For instance, with 128 addresses stored in a 64-bin hash table, the vast majority of the hash table bits are set, preventing only a small fraction of frames from reaching memory. In such instances, an external CAM is advised if the extra bus use cannot be tolerated. See Section 39.8, “CAM Interface.”

NOTE

The hash tables cannot be used to reject frames that match a set of selected addresses because unintended addresses can map to the same bit in the hash table. Thus, an external CAM must be used to implement this function.

39.15 Interpacket Gap Time

The minimum interpacket gap time for back-to-back transmission is 96 serial clocks. The receiver receives back-to-back frames with this minimum spacing. In addition, after the backoff algorithm, the transmitter waits for carrier sense to be negated before retransmitting the frame. The retransmission begins 96 serial clocks after carrier sense is negated if it stays negated for at least 60 serial clocks. So if there is no change in the carrier sense indication during the first 60 serial clocks after the retransmission begins 96 clocks after carrier sense is first negated

39.16 Handling Collisions

If a collision occurs during frame transmission, the Ethernet controller continues transmission for at least 32-bit times, transmitting a jam pattern of 32 ones. If the collision occurs during the preamble sequence, the jam pattern is sent after the sequence ends.

If a collision occurs within 64 byte times, the process is retried. The transmitter waits a random number of slot times. (A slot time is 512 bit times.) If a collision occurs after 64 byte times, no retransmission is performed, FCCE[TXE] is set, and the buffer is closed with a late-collision error indication in TxBD[LC]. If a collision occurs during frame reception, reception is stopped. This error is reported only in the RxBD if the frame is at least as long as the MINFLR or if FPSMR[RSH] = 1.

39.17 Internal and External Loopback

Both internal and external loopback are supported by the Ethernet controller. In loopback mode, both receive and transmit FIFO buffers are used and the FCC operates in full-duplex. Both internal and external loopback are configured using combinations of FPSMR[LPB]

and GFMR[DIAG]. Because of the full-duplex nature of the loopback operation, the performance of the other FCCs is degraded.

Internal loopback disconnects the FCC from the SI. The receive data is connected to the transmit data. The transmitted data from the transmit FIFO is received immediately into the receive FIFO. There is no heartbeat check in this mode.

In external loopback operation, the Ethernet controller listens for data received from the PHY while it is sending.

39.18 Ethernet Error-Handling Procedure

The Ethernet controller reports frame reception and transmission error conditions using the channel BDs, the error counters, and the FCC event register.

Transmission errors are described in Table 39-7.

Table 39-7. Transmission Errors

Error	Response
Transmitter underrun	The controller sends 32 bits that ensure a CRC error, terminates buffer transmission, closes the buffer, sets TxBD[UN] and FCCE[TXE]. The controller resumes transmission after receiving the RESTART TRANSMIT command.
Carrier sense lost during frame transmission	If no collision is detected in the frame, the controller sets TxBD[CSL] and FCCE[TXE], and it continues the buffer transmission normally. No retries are performed as a result of this error.
Retransmission attempts limit expired	The controller terminates buffer transmission, closes the buffer, sets TxBD[RL] and FCCE[TXE]. Transmission resumes after receiving the RESTART TRANSMIT command.
Late collision	The controller terminates buffer transmission, closes the buffer, sets TxBD[LC] and FCCE[TXE]. The controller resumes transmission after receiving the RESTART TRANSMIT command. Note that late collision parameters are defined in FPSMR[LCW].

Reception errors are described in Table 39-8.

Table 39-8. 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 writes the received data byte to the internal FIFO buffer over the previously received byte. The previous data byte and frame status are lost. The controller closes the buffer, sets RxBD[OV] and FCCE[RXF], and increments the discarded frame counter (DISFC). The receiver then enters hunt mode.
Busy error	A frame is received and discarded due to a lack of buffers. The controller sets FCCE[BSY] and increments the discarded frame counter (DISFC).

Table 39-8. Reception Errors

Error	Description
Non-octet error (dribbling bits)	The Ethernet controller handles a nibble of dribbling bits when the receive frame terminates as nonoctet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, the frame nonoctet aligned (RxBD[NO]) error is reported, FCCE[RXF] is set, and the alignment error counter (ALEC) in the parameter RAM is incremented. If there is no CRC error, no error is reported.
CRC error	When a CRC error occurs, the controller closes the buffer, and sets RxBD[CR] and FCCE[RXF]. Also, the CRC error counter (CRCEC) in the parameter RAM is incremented. After receiving a frame with a CRC error, the receiver enters hunt mode. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

39.18.1FCC Ethernet Mode Register (FPSMR)

The MPC2560 supports 10/100 Mbps Ethernet through a RMII interface (according to RMII Specification March 20, 1998). The RMII use a single reference clock (50 MHz) and seven pins which are a proper subset of the MII interface pins. Ethernet features are unchanged in RMII mode. To select RMII-PHY interface, a mode bit in the Ethernet mode register (FPSMR) has been added, as shown in Figure 39-6.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	HBC	FC	SBT	LPB	LCW	FDE	MON	—		PRO	FCE	RSH	—		RMII	—
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_1304 (FPSMR1), 0x9_1324 (FPSMR2), 0x9_1344 (FPSMR3)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	—					CAM	BRO	—	CRC		—					
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_1306 (FPSMR1), 0x9_1326 (FPSMR2), 0x9_1346 (FPSMR3)															

Figure 39-6. FCC Ethernet Mode Register (FPSMR*x*)

Table 8-2 describes FPSMR fields.

Table 39-9. FPSMR Ethernet Field Descriptions

Bits	Name	Description
0	HBC	Heartbeat checking 0 No heartbeat checking is performed. Do not wait for a collision after transmission. 1 Wait 40 transmit serial clocks for a collision asserted by the transceiver after transmission. TxBD[HB] is set if the heartbeat is not heard within 40 transmit serial clocks.
1	FC	Force collision 0 Normal operation 1 The channel forces a collision on transmission of every transmit frame. The MPC8560 should be configured in loopback operation when using this feature, which allows the user to test the MPC8280 collision logic. It causes the retry limit to be exceeded for each transmit frame.
2	SBT	Stop backoff timer 0 The backoff timer functions normally. 1 The backoff timer (for the random wait after a collision) is stopped whenever carrier sense is active. In this method, the retransmission is less aggressive than the maximum allowed in the IEEE 802.3 standard. The persistence (P_PER) feature in the parameter RAM can be used in combination with the SBT bit (or in place of the SBT bit).
3	LPB	Loopback operation 0 Normal operation (receiver does not receive when transmitter sends). 1 The channel is configured for internal or external loopback operation as determined by GFMR[DIAG]. For external loopback, configure DIAG for normal operation; for internal loopback configure DIAG for loopback operation.
4	LCW	Late collision window 0 A late collision is any collision that occurs at least 64 bytes from the preamble. 1 A late collision is any collision that occurs at least 56 bytes from the preamble.
5	FDE	Full duplex Ethernet 0 Disable full-duplex 1 Enable full-duplex. Must be set if FSMR[LPB] is set or external loopback is performed.
6	MON	RMON mode 0 Disable RMON mode 1 Enable RMON mode
7–8	—	Reserved, should be zero
9	PRO	Promiscuous 0 Check the destination address of incoming frames. 1 Receive the frame regardless of its address. A CAM can be used for address filtering when FSMR[CAM] is set.
10	FCE	Flow control enable 0 Flow control is not enabled 1 Flow control is enabled
11	RSH	Receive short frames 0 Discard short frames (frames smaller than the value specified in MINFLR). 1 Receive short frames.
12–13	—	Reserved, should be zero.
14	RMII	RMII interface mode 0 MII interface 1 RMII interface. RMII to/from MII conversion logic is enabled.
15–20	—	Reserved, should be zero.

Table 39-9. FPSMR Ethernet Field Descriptions (continued)

Bits	Name	Description
21	CAM	CAM address matching 0 Normal operation. 1 Use the CAM for address matching; CAM result (16 bits) is added at the end of the frame.
22	BRO	Broadcast address 0 Receive all frames containing the broadcast address. 1 Reject all frames containing the broadcast address unless FSMR[PRO] = 1.
23	—	Reserved, should be zero
24–25	CRC	CRC selection 0x Reserved. 10 32-bit CCITT-CRC (Ethernet). X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1. Select this to comply with Ethernet specifications. 11 Reserved.
26–31	—	Reserved, should be zero

39.18.2 Ethernet Event Register (FCCE)/Mask Register (FCCM)

The FCCE, shown in Figure 39-7, is used as the Ethernet event register when the FCC functions as an Ethernet controller. It generates interrupts and reports events recognized by the Ethernet channel. On recognition of an event, the Ethernet controller sets the corresponding FCCE bit. Interrupts generated by this register can be masked in the Ethernet mask register (FCCM).

The FCCM has the same bit format as FCCE. Setting an FCCM bit enables and clearing a bit masks the corresponding interrupt in the FCCE.

The FCCE can be read at any time. Bits are cleared by writing ones; writing zeros does not affect bit values. Unmasked FCCE bits must be cleared before the CP clears the internal interrupt request.

	0 – 7	8	9	10	11	12	13	14	15
Field	—	GRA	RXC	TXC	TXE	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000								
R/W	R/W								
Addr	0x9_1310 (FCCE1), 0x9_1330 (FCCE2), 0x9_1350 (FCCE3)/ 0x9_1314 (FCCM1), 0x9_1334 (FCCM2), 0x9_1354 (FCCM3)								

	16 – 31
Field	—
Reset	0000_0000_0000_0000
R/W	R/W
Addr	0x9_1312 (FCCE1), 0x9_1332 (FCCE2), 0x9_1352 (FCCE3)/ 0x9_1316 (FCCM1), 0x9_1336 (FCCM2), 0x9_1356 (FCCM3)

Figure 39-7. Ethernet Event Register (FCCE)/Mask Register (FCCM)

Table 39-10 describes FCCE/FCCM fields.

Table 39-10. FCCE/FCCM Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8	GRA	Graceful stop complete. A graceful stop, initiated by the GRACEFUL STOP TRANSMIT command, is complete. When the command is issued, GRA is set as soon the transmitter finishes sending a frame in progress. If no frame is in progress, GRA is set immediately.
9	RXC	RX control. A control frame has been received (FSMR[FCE] must be set). As soon as the transmitter finishes sending the current frame, a pause operation is performed.
10	TXC	TX control. An out-of-sequence frame was sent.
11	TXE	Tx error. An error occurred on the transmitter channel.
12	RXF	Rx frame. Set when a complete frame is received on the Ethernet channel.
13	BSY	Busy condition. Set when a frame is received and discarded due to a lack of buffers.
14	TXB	Tx buffer. Set when a buffer has been sent on the Ethernet channel.
15	RXB	Rx buffer. A buffer that was not a complete frame is received on the Ethernet channel.
16–31	—	Reserved, should be cleared.

Figure 39-8 shows interrupts that can be generated in the Ethernet protocol.

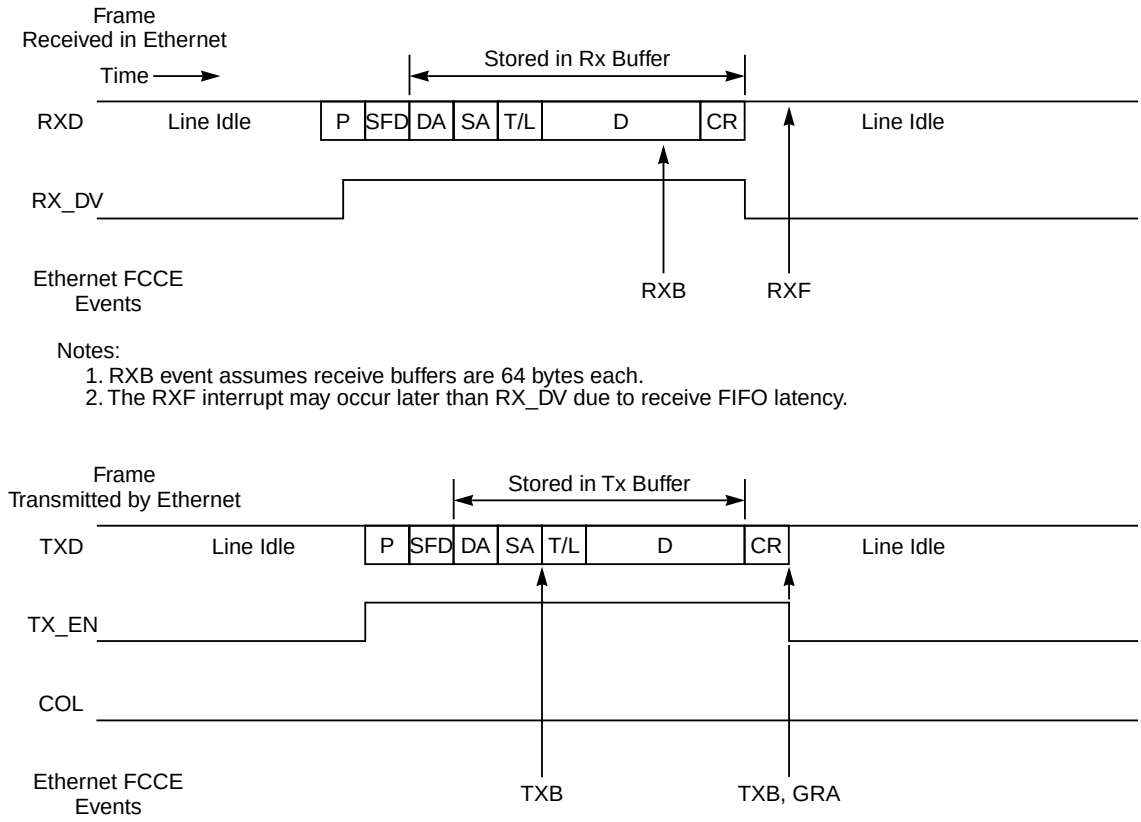


Figure 39-8. Ethernet Interrupt Events Example

Note that the FCC status register is not valid for the Ethernet protocol. The current state of the MII signals can be read through the parallel ports.

39.19 Ethernet RxBDs

The Ethernet controller uses the RxBD to report information about the received data for each buffer. Figure 39-9 shows the FCC Ethernet RxBD format.

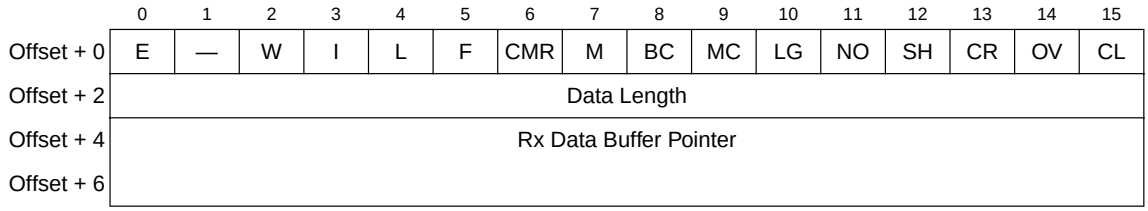


Figure 39-9. Fast Ethernet Receive Buffer (RxBD)

Table 39-11 describes Ethernet RxBD fields.

Table 39-11. RxBD Field Descriptions

Bits	Name	Description
0	E	Empty 0 The buffer associated with this RxBD is full or reception terminated due to an error. The core can examine or read to any fields of this RxBD. The CP does not use this BD as long as E = 0. 1 The associated buffer is empty. The RxBD and buffer are owned by the CP. Once E = 1, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared.
2	W	Wrap (final BD in RxBD table) 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to in the table. The number of RxBDs in this table is programmable and determined only by the W bit. The RxBD table must contain more than one BD in Ethernet mode.
3	I	Interrupt 0 No interrupt is generated after this buffer is used. 1 FCCE[RXB] or FCCE[RXF] are set when this buffer is used by the Ethernet controller. These two bits can cause interrupts if they are enabled.
4	L	Last in frame. Set by the Ethernet controller when this buffer is the last in a frame. This implies the end of the frame or a reception error, in which case one or more of the CL, OV, CR, SH, NO, and LG bits are set. The Ethernet controller writes the number of frame octets to the data length field. 0 Not the last buffer in a frame. 1 Last buffer in a frame.
5	F	First in frame. Set by the Ethernet controller when this buffer is the first in a frame. 0 Not the first buffer in a frame. 1 First buffer in a frame.
6	CMR	CAM match result for the frame. Set by the Ethernet controller when using a CAM for address matching and FPSMR[ECM] = 1. Valid only if the L bit is set. 0 A hit in the CAM. 1 A miss in the CAM.
7	M	Miss. Set by the Ethernet controller for frames that are accepted in promiscuous mode, but are flagged as a miss by the internal address recognition. Thus, while using promiscuous mode, the user uses the miss bit to determine quickly whether the frame is destined for this station. Valid only if RxBD[I] is set. 0 The frame is received because the address is recognized. 1 The frame is received because of promiscuous mode (address is not recognized).
8	BC	Broadcast address. Valid only for the last buffer in a frame (RxBD[L] = 1). The received frame address is the broadcast address.

Table 39-11. RxBD Field Descriptions (continued)

Bits	Name	Description
9	MC	Multicast address. Valid only for the last buffer in a frame (RxBD[L] = 1). The received frame address is a multicast address other than a broadcast address.
10	LG	Rx frame length violation. A frame length greater than the MFLR (maximum frame length) defined for this FCC is recognized.
11	NO	Rx nonoctet aligned frame. A frame that contained a number of bits not divisible by eight is received and the CRC check at the preceding byte boundary generated an error.
12	SH	Short frame. A frame length less than the MINFLR (minimum frame length) defined for this channel is recognized. This indication is possible only if the FPSMR[RSH] = 1.
13	CR	Rx CRC error. This frame contains a CRC error.
14	OV	Overrun. A receiver overrun occurred during frame reception.
15	CL	Collision. This frame is closed because a collision occurred during frame reception. Set only if a late collision occurs or if FPSMR[RSH] is set. The late collision definition is determined by the setting of FPSMR[LCW].

Data length is the number of octets the CP writes into this BD data buffer. It is written by the CP as the buffer is closed. When this BD is the last BD in the frame (RxBD[L] = 1), the data length contains the total number of frame octets (including four bytes for CRC). Note that at least as much memory should be allocated for each receive buffer as the size specified in MRBLR. MRBLR should be divisible by 32 and not less than 64.

The receive buffer pointer, which points to the first location of the associated data buffer, can reside in external memory. This value must be divisible by 32.

When a received frame's data length is an exact multiple of MRBLR, the last BD contains only the status and total frame length.

Note that at least two BDs must be prepared before beginning reception.

Figure 39-10 shows how RxBDs are used during Ethernet reception.

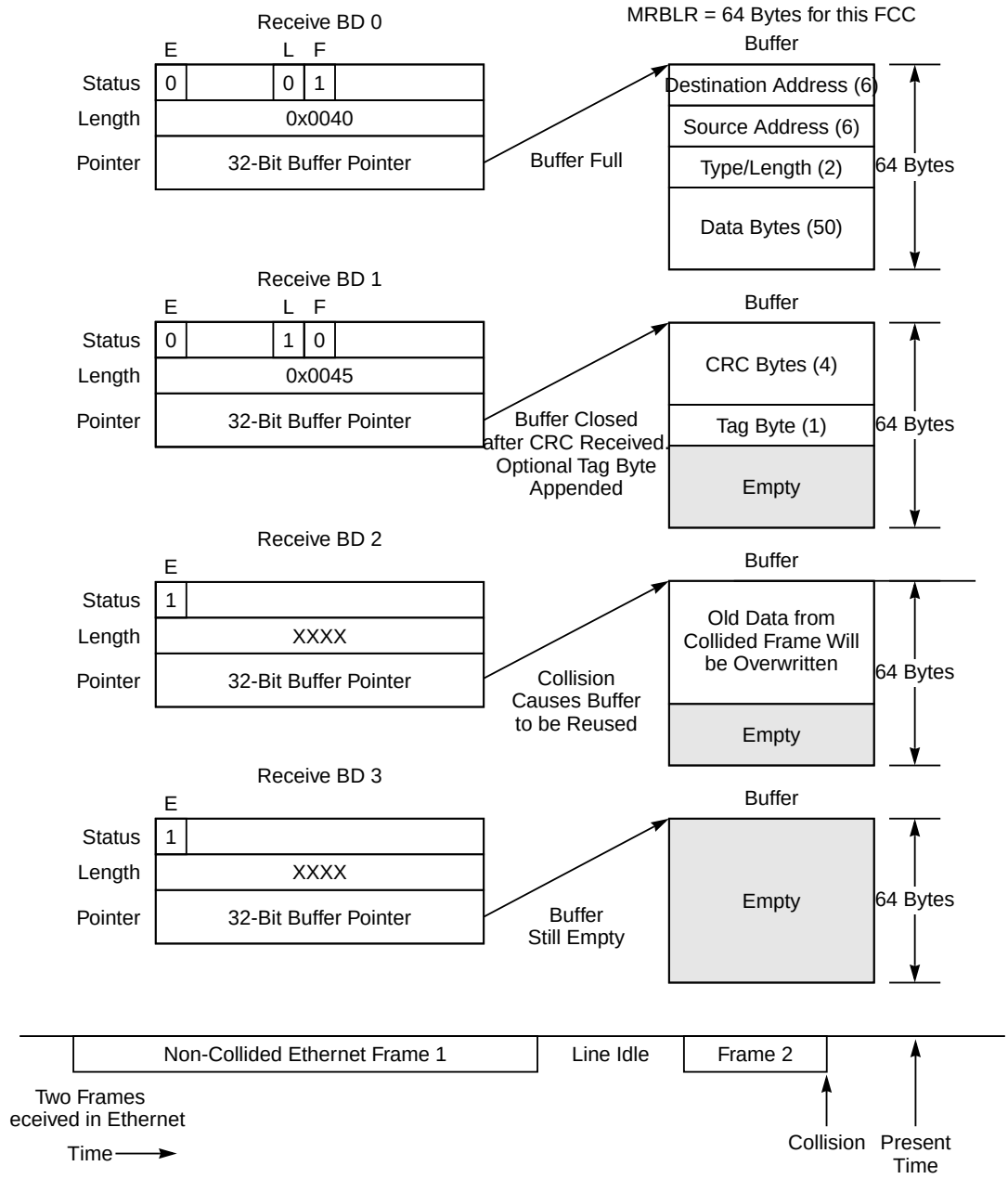


Figure 39-10. Ethernet Receiving Using RxBDs

39.20 Ethernet TxBDs

Data is sent to the Ethernet controller for transmission on an FCC channel by arranging it in buffers referenced by the channel's TxBD table. The Ethernet controller uses TxBDs to confirm transmission or indicate errors so the core knows when buffers have been serviced. Figure 39-11 shows the FCC Ethernet TxBD format.

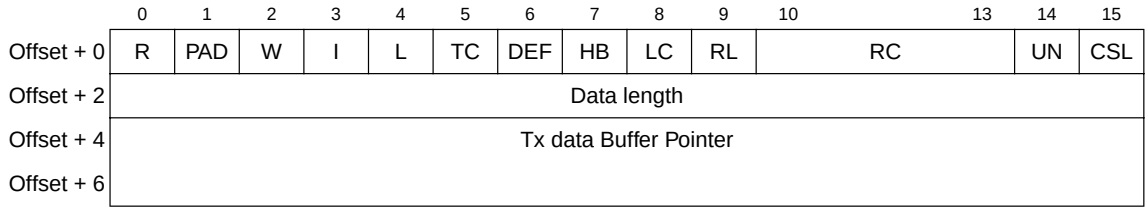


Figure 39-11. Fast Ethernet Transmit Buffer (TxBD)

Table 39-12 describes Ethernet TxBD fields.

Table 39-12. Ethernet TxBD Field Definitions

Field	Name	Description
0	R	Ready 0 The buffer associated with this BD is not ready for transmission; the user can manipulate this BD or its associated buffer. The CP clears R after the buffer has been sent or after an error. 1 The buffer is ready to be sent. The buffer is either waiting or in the process of being sent. The user cannot change fields in this BD or its associated buffer once R = 1.
1	PAD	Short frame padding. Valid only when L = 1; otherwise, it is ignored. 0 Do not add PADs to short frames. 1 Add PADs to short frames. PAD bytes are inserted until the length of the transmitted frame equals the MINFLR. The PAD bytes are stored in a buffer pointed to by PAD_PTR in the parameter RAM.
2	W	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer is used, the CP receives incoming data into the first BD that TBASE points to in the table. The number of TxBDs in this table is programmable and determined only by the W bit. The TxBD table must contain more than one BD in Ethernet mode.
3	I	Interrupt 0 No interrupt is generated after this buffer is serviced. 1 FCCE[TXB] or FCCE[TXE] is set after this buffer is serviced. These bits can cause interrupts if they are enabled.
4	L	Last 0 Not the last buffer in the transmit frame. 1 Last buffer in the current transmit frame.
5	TC	Tx CRC. Valid only when the L bit is set; otherwise, it is ignored. 0 End transmission immediately after the last data byte. 1 Transmit the CRC sequence after the last data byte.
6	DEF	Defer indication. This frame did not have a collision before it was sent but it was sent late because of deferring.
7	HB	Heartbeat. The collision input is not asserted within 40 transmit serial clocks following completion of transmission. This bit cannot be set unless FPSMR[HBC] = 1. Written by the Ethernet controller after sending the associated buffer.
8	LC	Late collision. A collision occurred after the number of bytes defined in FPSMR[LCW] (56 or 64) are sent. The Ethernet controller terminates the transmission and updates LC after sending the buffer.
9	RL	Retransmission limit. The transmitter failed (RET_LIM + 1) attempts to successfully send a message due to repeated collisions. The Ethernet controller updates RL after sending the buffer.

Table 39-12. Ethernet TxBD Field Definitions (continued)

Field	Name	Description
10–13	RC	Retry count. Indicates the number of retries required for this frame to be successfully sent. If RC = 0, the frame is sent correctly the first time. If RC = 15 and RET_LIM = 15 in the parameter RAM, 15 retries were needed. If RC = 15 and RET_LIM > 15, 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
14	UN	Underrun. The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer. The Ethernet controller updates UN after sending the buffer.
15	CSL	Carrier sense lost. Carrier sense is lost during frame transmission. The Ethernet controller updates CSL after sending the buffer.

Data length is the number of octets the Ethernet controller should transmit from this BD data buffer. This value should be greater than zero. The CP never modifies the data length in a TxBD.

Tx data buffer pointer, which contains the address of the associated data buffer, can be even or odd. The buffer can reside in external memory. The CP never modifies the buffer pointer.

Chapter 40
FCC HDLC Controller

Layer 2 of the seven-layer OSI model is the data link layer (DLL), in which HDLC is one of the most common protocols. The framing structure of HDLC is shown in Figure 40-1. HDLC uses a zero insertion/deletion process (commonly known as bit stuffing) to ensure that the bit pattern of the delimiter flag does not occur in the fields between flags. The HDLC frame is synchronous and therefore relies on the physical layer for a method of clocking and of synchronizing the transmitter/receiver.

Because the layer 2 frame can be transmitted over a point-to-point link, a broadcast network, or a packet-and-circuit switched system, an address field is needed for the frame's destination address. The length of this field is commonly 0, 8, or 16 bits, depending on the data link layer protocol. For instance, SDLC and LAPB use an 8-bit address and SS#7 has no address field because it is used always in point-to-point signaling links. LAPD further divides its 16-bit address into different fields to specify various access points within one device. It also defines a broadcast address. Some HDLC-type protocols also permit extended addressing beyond 16 bits.

The 8- or 16-bit control field provides a flow-control number and defines the frame type (control or data). The exact use and structure of this field depends upon the protocol using the frame. Data is transmitted in the data field, which can vary in length depending upon the protocol using the frame. Layer 3 frames are carried in this data field.

Error control is implemented by appending a cyclic redundancy check (CRC) to the frame, which in most protocols is 16-bits long but can be as long as 32-bits. In HDLC, the lsb of each octet is transmitted first and the msb of the CRC is transmitted first.

When GFMR[MODE] selects HDLC mode, that FCC functions as an HDLC controller. When an FCC in HDLC mode is used with a nonmultiplexed modem interface, the FCC outputs are connected directly to the external pins. Modem signals can be supported through the appropriate port pins. The receive and transmit clocks can be supplied either externally or from the bank of baud-rate generators. The HDLC controller can also be connected to one of the TDM channels of the serial interface and used with the TSA. The HDLC controller consists of separate transmit and receive sections whose operations are asynchronous with the core and can either be synchronous or asynchronous with other FCCs. The user can allocate external buffer descriptors (BDs) for receive and transmit tasks so many frames can be sent or received without core intervention.

40.1 Key Features

Key features of the HDLC include the following:

- Flexible data buffers with multiple buffers per frame
- Separate interrupts for frames and buffers (receive and transmit)
- Received frames threshold to reduce interrupt overhead
- Four address comparison registers with masks
- Maintenance of four 16-bit error counters
- Flag/abort/idle generation and detection
- Zero insertion/deletion
- 16- or 32-bit CRC-CCITT generation/checking
- Detection of nonoctet-aligned frames
- Detection of frames that are too long
- Programmable flags (0–15) between successive frames
- External BD table
- Up to T3 rate
- Support of time stamp mode for Rx frames
- Support of nibble mode HDLC (4 bits per clocks)

40.2 HDLC Channel Frame Transmission Processing

The HDLC transmitter is designed to work with almost no core intervention. When the core enables a transmitter, it starts sending flags or idles as programmed in the HDLC mode register (FPSMR). The HDLC controller polls the first BD in the transmit channel BD table. When there is a frame to transmit, the HDLC controller fetches the data (address, control, and information) from the first buffer and begins sending the frame after first inserting the user-specified minimum number of flags between frames. When the end of the current buffer is reached and TxBD[L] (last buffer in frame) is set, the FCC appends the CRC (if selected) and closing flag. In HDLC, the lsb of each octet and the msb of the CRC are sent first. Figure 40-1 shows a typical HDLC frame.

Opening Flag	Address	Control	Information (Optional)	CRC	Closing Flag
8 Bits	16 Bits	8 Bits	8*n* Bits	16 Bits	8 Bits

Figure 40-1. HDLC Framing Structure

After the closing flag is sent, the HDLC controller writes the frame status bits into the BD and clears the R bit. When the end of the current BD is reached and the L (last) bit is not set (working in multibuffer mode), only the R bit is cleared. In either mode, an interrupt can

be issued if the I bit in the TxBD is set. The HDLC controller then proceeds to the next TxBD in the table. In this way, the core can be interrupted after each buffer, after a specific buffer, after each frame, or after a number of frames.

To rearrange the transmit queue before the CP has sent all buffers, issue the STOP TRANSMIT command. This can be useful for sending expedited data before previously linked buffers or for error situations. When receiving the STOP TRANSMIT command, the HDLC controller aborts the current frame transmission and starts transmitting idles or flags. When the HDLC controller is given the RESTART TRANSMIT command, it resumes transmission. To insert a high-priority frame without aborting the current frame, the GRACEFUL STOP TRANSMIT command can be issued. A special interrupt (GRA) can be generated in the event register when the current frame is complete.

40.3 HDLC Channel Frame Reception Processing

The HDLC receiver is designed to work with almost no core intervention and can perform address recognition, CRC checking, and maximum frame length checking. The received frame is available for any HDLC-based protocol. When the core enables a receiver, the receiver waits for an opening flag character. When it detects the first byte of the frame, the HDLC controller compares the frame address against the user-programmable addresses. The user has four 16-bit address registers and an address mask available for address matching. The HDLC controller compares the received address field to the user-defined values after masking with the address mask. The HDLC controller can also detect broadcast (all ones) address frames if one address register is written with all ones.

If a match is detected, the HDLC controller checks the prefetched BD; if it is empty, it starts transferring the incoming frame to the BD's associated buffer. When the buffer is full, the HDLC controller clears BD[E] and generates an interrupt if BD[I] = 1. If the incoming frame is larger than the buffer, the HDLC controller fetches the next BD in the table and, if it is empty, continues transferring the frame to the associated buffer.

During this process, the HDLC controller checks for frames that are too long. When the frame ends, the CRC field is checked against the recalculated value and written to the buffer. The data length written to the last BD in the HDLC frame is the length of the entire frame. This enables HDLC protocols that lose frames to correctly recognize a frame-too-long condition.

The HDLC controller then sets the last buffer in frame bit, writes the frame status bits into the BD, and clears the E bit and fetches the next BD. The HDLC controller then generates a maskable interrupt, indicating that a frame was received and is in memory. The HDLC controller then waits for a new frame. Back-to-back frames can be received separated only by a single shared flag.

The user can configure the HDLC controller not to interrupt the core until a specified number of frames have been received. This is configured in the received frames threshold

(RFTHR) location of the parameter RAM. This function can be combined with a timer to implement a time-out if fewer than the threshold number of frames are received.

40.4 HDLC Parameter RAM

When an FCC operates in HDLC mode, the protocol-specific area of the FCC parameter RAM is mapped with the HDLC-specific parameters in Table 40-1.

Table 40-1. FCC HDLC-Specific Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x3C	—	3 Words	Reserved
0x44	C_MASK	Word	CRC constant. For the 16-bit CRC-CCITT, initialize C_MASK to 0x0000_F0B8. For the 32-bit CRC-CCITT, initialize C_MASK to 0xDEBB_20E3.
0x48	C_PRES	Word	CRC preset. For the 16-bit CRC-CCITT, initialize C_PRES to 0x0000_FFFF. For the 32-bit CRC-CCITT, initialize C_PRES to 0xFFFF_FFFF.
0x4C	DISFC [2]	Hword	Discard frame counter. Counts error-free frames discarded due to lack of buffers.
0x4E	CRCEC[2]	Hword	CRC error counter. Counts frames not addressed to the user or frames received in the BSY condition, but does not include overrun, $\overline{CD}$ lost, or abort errors.
0x50	ABTSC[2]	Hword	Abort sequence counter
0x52	NMARC[2]	Hword	Nonmatching address Rx counter. Counts nonmatching addresses received (error-free frames only). See the HMASK and HADDR[1–4] parameter description.
0x54	MAX_CNT	Word	Max_length counter. Temporary decrementing counter that tracks frame length.
0x58	MFLR	Hword	Max frame length register. If the HDLC controller detects an incoming HDLC frame that exceeds the user-defined value in MFLR, the rest of the frame is discarded and the LG (Rx frame too long) bit is set in the last BD belonging to that frame. The HDLC controller waits for the end of the frame and then reports the frame status and length in the last RxBD. MFLR includes all in-frame bytes between the opening and closing flags (address, control, data, and CRC).
0x5A	RFTHR	Hword	Received frames threshold. Used to reduce the interrupt overhead that might otherwise occur when a series of short HDLC frames arrives, each causing an RXF interrupt. By programming RFTHR, the user lowers the frequency of RXF interrupts, which occur only when the RFTHR value is reached. Note that the user should provide enough empty RxBDs to receive the number of frames specified in RFTHR.
0x5C	RFCNT	Hword	Received frames count. A decrementing counter used to implement this feature. Initialize this counter with RFTHR.
0x5E	HMASK	Hword	HMASK and HADDR[1–4]. The HDLC controller reads the frame address from the HDLC receiver, checks it against the four address register values, and masks the result with HMASK. In HMASK, a 1 represents a bit position for which address comparison should occur; 0 represents a masked bit position. When addresses match, the address and subsequent data are written into the buffers. When addresses do not match and the frame is error-free, the nonmatching address received counter (NMARC) is incremented. Note that for 8-bit addresses, mask out (clear) the eight high-order bits in HMASK. The eight low-order bits and HADDRx should contain the address byte that immediately follows the opening flag. For example, to recognize a frame that begins 0x7E (flag), 0x68, 0xAA, using 16-bit address recognition, HADDRx should contain 0xAA68 and HMASK should contain 0xFFFF. See Figure 40-2.
0x60	HADDR1	Hword	
0x62	HADDR2	Hword	
0x64	HADDR3	Hword	
0x66	HADDR4	Hword	

Table 40-1. FCC HDLC-Specific Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x68	TS_TMP	Hword	Temporary storage
0x6A	TMP_MB	Hword	Temporary storage

[1] Offset from FCC base: 0x8400 (FCC1), 0x8500 (FCC2) and 0x8600 (FCC3); see Section 20.5.2, “Parameter RAM.”

[2] DISFC, CRCEC, ABTSC, and NMARC—These 16-bit (modulo 216) counters are maintained by the CP. The user can initialize them while the channel is disabled.

Figure 40-2 shows an example of using HMASK and HADDR[1–4].

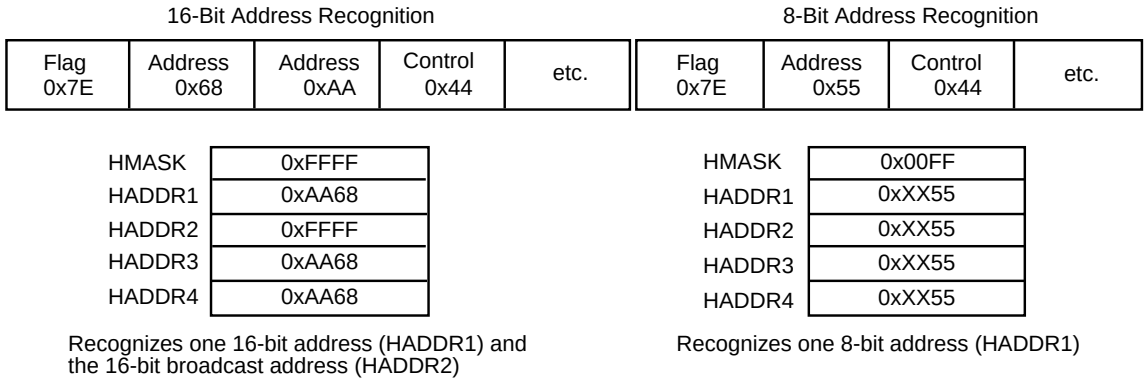


Figure 40-2. HDLC Address Recognition Example

40.5 Programming Model

The core configures each FCC to operate in the protocol specified in GFMR[MODE]. The HDLC controller uses the same data structure as other modes. This data structure supports multibuffer operation and address comparisons.

40.5.1 HDLC Command Set

The transmit and receive commands are issued to the CPCR; see Section 20.4, “Command Set.”

Table 40-2 describes the transmit commands that apply to the HDLC controller.

Table 40-2. Transmit Commands

Command	Description
STOP TRANSMIT	After the hardware or software is reset and the channel is enabled in the FCC mode register, the channel is in transmit enable mode and starts polling the first BD in the table every 256 transmit clocks (immediately if FTODR[TOD] = 1). STOP TRANSMIT command disables the transmission of frames on the transmit channel. If this command is received by the HDLC controller during frame transmission, transmission is aborted after a maximum of 64 additional bits are sent and the transmit FIFO buffer is flushed. The TBPTR is not advanced, no new BD is accessed, and no new frames are sent for this channel. The transmitter sends an abort sequence consisting of 0x7F (if the command was given during frame transmission) and begins sending flags or idles, as indicated by the HDLC mode register. Note that if FPSMR[MFF] = 1, one or more small frames can be flushed from the transmit FIFO buffer. The GRACEFUL STOP TRANSMIT command can be used to avoid this.
GRACEFUL STOP TRANSMIT	Used to stop transmission smoothly rather than abruptly, as performed by the regular STOP TRANSMIT command. It stops transmission after the current frame finishes sending or immediately if no frame is being sent. FCCE[GRA] is set once transmission has stopped. Then the HDLC transmit parameters (including BDs) can be modified. The TBPTR points to the next TxBD in the table. Transmission begins once the R bit of the next BD is set and the RESTART TRANSMIT command is issued.
RESTART TRANSMIT	Enables character transmission on the transmit channel. This command is expected by the HDLC controller after a STOP TRANSMIT command, after a STOP TRANSMIT command is issued and the channel in its FCC mode register is disabled, after a GRACEFUL STOP TRANSMIT command, or after a transmitter error (underrun or $\overline{\text{CTS}}$ lost with no automatic frame retransmission). The HDLC controller resumes sending from the current TBPTR in the channel TxBD table.
INIT TX PARAMETERS	Initializes all transmit parameters in this serial channel parameter RAM to their reset state. This command should only be issued when the transmitter is disabled. Notice that the INIT TX AND RX PARAMETERS command can also be used to reset the transmit and receive parameters.

Table 40-3 describes the receive commands that apply to the HDLC controller.

Table 40-3. Receive Commands

Command	Description
ENTER HUNT MODE	After the hardware or software is reset and the channel is enabled in the FCC mode register, the channel is in receive enable mode and uses the first BD in the table. The ENTER HUNT MODE command is generally used to force the HDLC receiver to abort reception of the current frame and enter the hunt mode. In hunt mode, the HDLC controller continually scans the input data stream for the flag sequence. After receiving the command, the current receive buffer is closed, the error status flags and length field are cleared, RxBD[E] (the empty bit) is set, and the CRC calculation is reset. Further frame reception uses the current RxBD.
INIT RX PARAMETERS	Initializes all the receive parameters in this serial channel parameter RAM to their reset state and should be issued only when the receiver is disabled. Notice that the INIT TX AND RX PARAMETERS command resets both receive and transmit parameters.

40.5.2 HDLC Error Handling

The HDLC controller reports frame reception and transmission error conditions using the channel BDs, error counters, and HDLC event register (FCCE). Table 40-4 describes HDLC transmission errors, which are reported through the TxBD.

Table 40-4. HDLC Transmission Errors

Error	Description
Transmitter Underrun	When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the underrun (U) bit in the BD, and generates the TXE interrupt if it is enabled. The channel resumes transmission after receiving the RESTART TRANSMIT command.
$\overline{\text{CTS}}$ Lost during Frame Transmission	When this error occurs, the channel terminates buffer transmission, closes the buffer, sets TxBD[CT], and generates a TXE interrupt (if it is enabled). The channel resumes transmission after receiving the RESTART TRANSMIT command.

Table 40-5 describes HDLC reception errors, which are reported through the RxBD.

Table 40-5. HDLC Reception Errors

Error	Description
Overrun Error	The HDLC controller maintains an internal FIFO buffer for receiving data. The CP begins programming the SDMA channel and updating the CRC whenever data is received in the FIFO buffer. When a receive FIFO overrun occurs, the channel writes the received data byte to the internal FIFO buffer over the previously received byte. The previous byte and the frame status are lost. The channel closes the buffer with RxBD[OV] set and generates the RXF interrupt if it is enabled. The receiver then enters hunt mode. Even if the overrun occurs during a frame whose address is not matched in the address recognition logic, an RxBD with data length two is opened to report the overrun and the RXF interrupt is generated if it is enabled.
$\overline{\text{CD}}$ Lost During Frame Reception	When this error occurs, the channel terminates frame reception, closes the buffer, sets RxBD[CD], and generates the RXF interrupt if it is enabled. This error has highest priority. The rest of the frame is lost and other errors are not checked in that frame. At this point, the receiver enters hunt mode.
Abort Sequence	The HDLC controller detects an abort sequence when seven or more consecutive ones are received. When this error occurs and the HDLC controller receives a frame, the channel closes the buffer by setting RxBD[AB] and generates the RXF interrupt (if enabled). The channel also increments the abort sequence counter. The CRC and nonoctet error status conditions are not checked on aborted frames. The receiver then enters hunt mode. When an abort sequence is received, the user is given no indication that an HDLC controller is not currently receiving a frame.
Nonoctet Aligned Frame	When this error occurs, the channel writes the received data to the data buffer, closes the buffer, sets the Rx nonoctet aligned frame bit RxBD[NO], and generates the RXF interrupt (if it is enabled). The CRC error status should be disregarded on nonoctet frames. After a nonoctet aligned frame is received, the receiver enters hunt mode. An immediate back-to-back frame is still received. The nonoctet data portion may be derived from the last byte in the buffer by finding the least-significant set bit, which marks the end of valid data as follows: msb | | | | | | | lsb Valid data | 1 | 0 | 0 | 0
CRC Error	When this error occurs, the channel writes the received CRC to the data buffer, closes the buffer, sets RxBD[CR], and generates the RXF interrupt (if it is enabled). The channel also increments the CRC error counter. After receiving a frame with a CRC error, the receiver enters hunt mode. An immediate back-to-back frame is still received. CRC checking cannot be disabled, but the CRC error can be ignored if checking is not required.

40.6 HDLC Mode Register (FPSMR)

When an FCC is configured for HDLC mode, the FPSMR is used as the HDLC mode register, shown in Figure 40-3.

Bits	0–3	4	5	6–8	9	10–15
Field	NOF	FSE	MFF	—	TS	—
Reset	0000_0000_0000_0000					
R/W	R/W					
Addr	0x9_1304 (FPSMR1), 0x9_1324 (FPSMR2), 0x9_1324 (FPSMR3)					

Bits	16	17–23	24–25	26–31
Field	NBL	—	CRC	—
Reset	0000_0000_0000_0000			
R/W	R/W			
Addr	0x9_1306 (FPSMR1), 0x9_1326 (FPSMR2), 0x9_1326 (FPSMR3)			

Figure 40-3. HDLC Mode Register (FPSMR)

The FPSMR fields are described in Table 40-6.

Table 40-6. FPSMR Field Descriptions

Bits	Name	Description
0–3	NOF	Number of flags. Minimum number of flags between or before frames (0–15 flags). If NOF = 0000, no flags are inserted between the frames. Thus, for back-to-back frames, the closing flag of one frame is immediately followed by the opening flag of the next frame.
4	FSE	Flag sharing enable. This bit is valid only if GFMR[RTSM] is set. 0 Normal operation 1 If NOF = 0000, a single shared flag is transmitted between back-to-back frames. Other values of NOF are decremented by 1 when FSE is set. This is useful in signaling system #7 applications.
5	MFF	Multiple frames in FIFO. Setting MFF applies only when in $\overline{\text{RTS}}$ mode (GFMR*x*[RTSM] = 1). 0 Normal operation. The transmit FIFO buffer must never contain more than one HDLC frame. The $\overline{\text{CTS}}$ lost status is reported accurately on a per-frame basis. The receiver is not affected by this bit. 1 The transmit FIFO buffer can contain multiple frames, but lost $\overline{\text{CTS}}$ is not guaranteed to be reported on the exact buffer/frame it occurred on. This option, however, can improve the performance of HDLC transmissions for small back-to-back frames or if the user prefers to strongly limit the number of flags sent between frames. MFF does not affect the receiver.
7–8	—	Reserved, should be cleared.
9	TS	Time stamp 0 Normal operation. 1 A 32-bit time stamp is added at the beginning of the receive BD data buffer, thus the buffer pointer must be (32-byte aligned - 4). The BD's data length does not include the time stamp. See Section 20.3.7, "RISC Time-Stamp Control Register (RTSCR)."
10–15	—	Reserved, should be cleared.

Table 40-6. FPSMR Field Descriptions (continued)

Bits	Name	Description
16	NBL	Nibble mode enable 0 Nibble mode disabled (1 bit of data per clock). Note that at the end of the frame (after the closing flag), $\overline{RTS}$ negates immediately after the active edge of TCLK. 1 Nibble mode enabled (4 bits of data per clock). Note that at the end of the frame (after the closing flag), $\overline{RTS}$ negates a maximum of 5 CPM clocks after the active edge of TCLK.
17–23	—	Reserved, should be cleared.
24-25	CRC	CRC selection 00 16-bit CCITT-CRC (HDLC). X16 + X12 + X5 + 1 01 Reserved 10 32-bit CCITT-CRC (Ethernet and HDLC). X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X1 +1 11 Reserved
26–31	—	Reserved, should be cleared.

40.7 HDLC Receive Buffer Descriptor (RxBD)

The HDLC controller uses the RxBD to report on data received for each buffer. Figure 40-4 shows an example of the RxBD process.

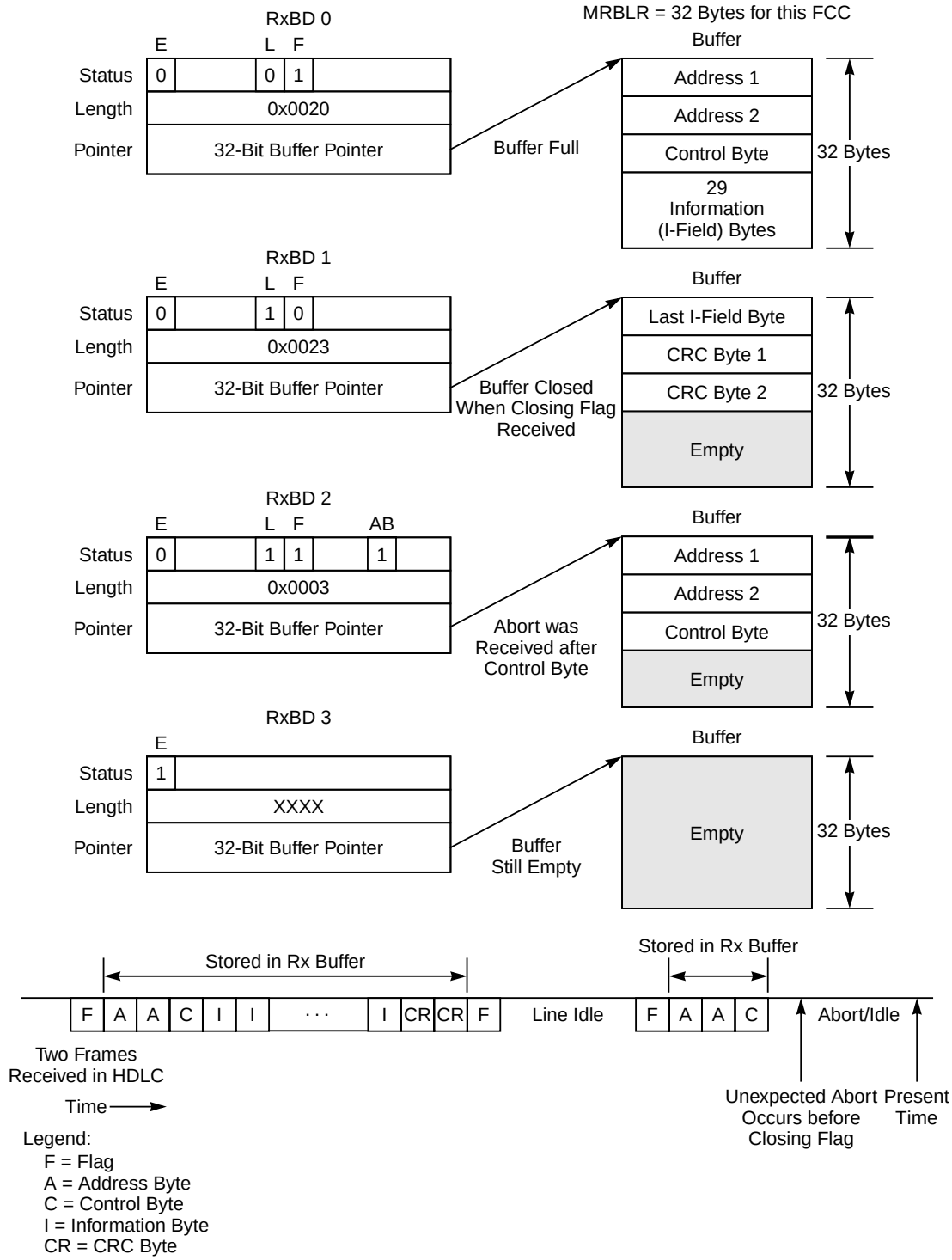


Figure 40-4. FCC HDLC Receiving Using RxBDs

Figure 40-5 shows the FCC HDLC RxBD.

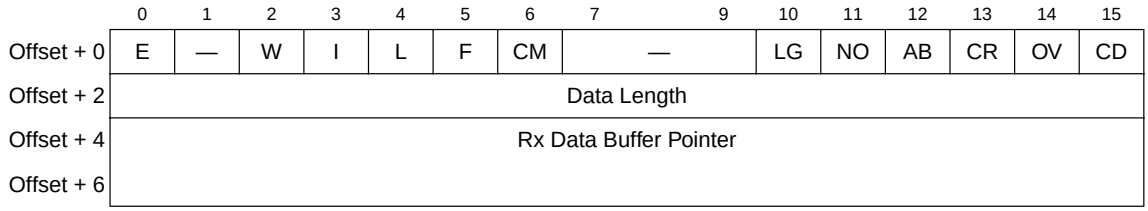


Figure 40-5. FCC HDLC Receive Buffer Descriptor (RxBD)

Table 40-7 describes RxBD fields.

Table 40-7. RxBD field Descriptions

Bits	Name	Description
0	E	Empty 0 The buffer is full with received data or data reception stopped because of an error. The core can read or write to any fields of this RxBD. The CP does not use this BD while E = 0. 1 The buffer associated with this BD is empty. This RxBD and its associated receive buffer are owned by the CP. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared.
2	W	Wrap (final BD in table) 0 Not the last BD in the RxBD table. 1 Last BD in the RxBD table. After this buffer is used, the CP receives incoming data into the first BD that RBASE points to in the table. The number of RxBDs in this table is programmable and is determined only by the W bit and the overall space constraints of the dual-port RAM. The RxBD table must contain more than one BD in HDLC mode.
3	I	Interrupt 0 The RXB bit is not set after this buffer is used, but RXF operation remains unaffected. 1 FCCE[RXB] or FCCE[RXF] is set when the HDLC controller uses this buffer. These two bits can cause interrupts if they are enabled.
4	L	Last in frame. Set by the HDLC controller when this buffer is the last one in a frame. This implies the reception of a closing flag or reception of an error, in which case one or more of the CD, OV, AB, and LG bits are set. The HDLC controller writes the number of frame octets to the data length field. 0 Not the last buffer in a frame. 1 Last buffer in a frame.
5	F	First in frame. Set by the HDLC controller when this buffer is the first in a frame. 0 Not the first buffer in a frame. 1 First buffer in a frame.
6	CM	Continuous mode 0 Normal operation. 1 The E bit is not cleared by the CP after this BD is closed, allowing the associated data buffer to be automatically overwritten the next time the CP accesses this BD. However, the E bit is cleared if an error occurs during reception, regardless of the CM bit.
7–9	—	Reserved, should be cleared.
10	LG	Rx frame length violation. A frame length greater than the maximum defined for this channel is recognized, and only the maximum-allowed number of bytes (MFLR) is written to the data buffer. This event is not reported until the RxBD is closed, the RXF bit is set, and the closing flag is received. The number of bytes received between flags is written to the data length field of this BD.

Table 40-7. RxBD field Descriptions (continued)

Bits	Name	Description
11	NO	Rx nonoctet-aligned frame. Set when a received frame contains a number of bits not divisible by eight.
12	AB	Rx abort sequence. At least seven consecutive 1s are received during frame reception.
13	CR	Rx CRC error. This frame contains a CRC error. Received CRC bytes are written to the receive buffer.
14	OV	Overrun. A receiver overrun occurs during frame reception.
15	CD	Carrier detect lost. $\overline{CD}$ has negated during frame reception. This bit is valid only for NMSI mode.

The RxBD status bits are written by the HDLC controller after receiving the associated data buffer.

The remaining RxBD parameters are as follows:

- Data length is the number of octets the CP writes into this BD's data buffer. It is written by the CP once the BD is closed. When this is the last BD in the frame (L = 1), this field contains the total number of frame octets, including 2 or 4 bytes for CRC. The memory allocated for this buffer should be no smaller than the MRBLR value.
- Rx data buffer pointer. The receive buffer pointer, which always points to the first location of the associated data buffer, resides in internal or external memory and must be divisible by 32 unless FPSMR[TS] = 1 (see Table 40-6).

40.8 HDLC Transmit Buffer Descriptor (TxBD)

Data is presented to the HDLC controller for transmission on an FCC channel by arranging it in buffers referenced by the channel TxBD table. The HDLC controller confirms transmission (or indicates errors) using the BDs to inform the core that the buffers have been serviced. Figure 40-6 shows the FCC HDLC TxBD.

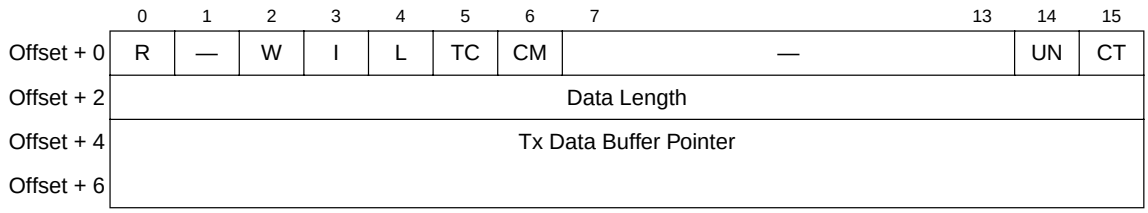


Figure 40-6. FCC HDLC Transmit Buffer Descriptor (TxBD)

Table 40-8 describes HDLC TxBD fields.

Table 40-8. HDLC TxBD Field Descriptions

Bits	Name	Description
0	R	Ready 0 The buffer associated with this BD is not ready for transmission. The user can manipulate this BD or its associated buffer. The CP clears R after the buffer has been sent or an error occurs. 1 The buffer is ready to be sent. The transmission may have begun, but it has not completed. The user cannot set fields in this BD once R is set.
1	—	Reserved, should be cleared.
2	W	Wrap (final BD in table) 0 Not the last BD in the TxBD table. 1 Last BD in the TxBD table. After this buffer has been used, the CP sends data from the first BD that TBASE points to in the table. The number of TxBDs in this table is determined only by the W bit and the overall space constraints of the dual-port RAM.
3	I	Interrupt 0 No interrupt is generated after this buffer is serviced. 1 Either FCCE[TXB] or FCCE[TXE] is set when this buffer is serviced by the HDLC controller. These bits can cause interrupts if they are enabled.
4	L	Last 0 Not the last buffer in the frame. 1 Last buffer in the current frame.
5	TC	Tx CRC.Valid only when the L bit is set. Otherwise, it is ignored. 0 Transmit the closing flag after the last data byte. This setting can be used to send a bad CRC after the data for testing purposes. 1 Transmit the CRC sequence after the last data byte.
6	CM	Continuous mode 0 Normal operation. 1 The R bit is not cleared by the CP after this BD is closed, allowing the buffer to be retransmitted automatically the next time the CP accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of the CM bit.
7–13	—	Reserved, should be cleared.
14	UN	Underrun. The HDLC controller encounters a transmitter underrun condition while sending the buffer. The HDLC controller writes UN after sending the buffer.
15	CT	$\overline{\text{CTS}}$ lost. Set when $\overline{\text{CTS}}$ is lost during frame transmission in NMSI mode. If data from more than one buffer is in the FIFO buffer when this error occurs, CT is set in the currently open TxBD. The HDLC controller writes CT after sending the buffer.

The TxBD status bits are written by the HDLC controller after sending the associated data buffer.

The remaining TxBD parameters are as follows:

- Data length is the number of bytes the HDLC controller should transmit from this data buffer; it is never modified by the CP. The value of this field should be greater than zero.
- Tx data buffer pointer. The transmit buffer pointer, which contains the address of the associated data buffer, can be even or odd. The buffer can reside in internal or external memory. This value is never modified by the CP.

40.9 HDLC Event Register (FCCE)/ Mask Register (FCCM)

The FCCE is used as the HDLC event register when the FCC operates as an HDLC controller. The FCCE reports events recognized by the HDLC channel and generates interrupts. On recognition of an event, the HDLC controller sets the corresponding FCCE bit. FCCE bits are cleared by writing ones; writing zeros does not affect bit values. All unmasked bits must be cleared before the CP clears the internal interrupt request.

Interrupts generated by the FCCE can be masked in the HDLC mask register (FCCM), which has the same bit format as FCCE. If an FCCM bit = 1, the corresponding interrupt in the event register is enabled. If the bit is 0, the interrupt is masked. Figure 40-7 shows the HDLC event register (FCCE) and the corresponding mask register (FCCM).

	0–7	8	9–10	11	12	13	14	15
Field	—	GRA	—	TXE	RXF	BSY	TXB	RXB
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	0x9_1310 (FCCE1), 0x9_1330 (FCCE2), 0x9_1350 (FCCE3)/ 0x9_1314 (FCCM1), 0x9_1334 (FCCM2), 0x9_1354 (FCCM3)							

	16–21	22	23	24–31
Field	—	FLG	IDL	—
Reset	0000_0000_0000_0000			
R/W	R/W			
Addr	0x9_1312 (FCCE1), 0x9_1332 (FCCE2), 0x9_1352 (FCCE3)/ 0x9_1316 (FCCM1), 0x9_1336 (FCCM2), 0x9_1356 (FCCM3)			

Figure 40-7. HDLC Event Register (FCCE)/Mask Register (FCCM)

Table 40-9 describes FCCE/FCCM fields.

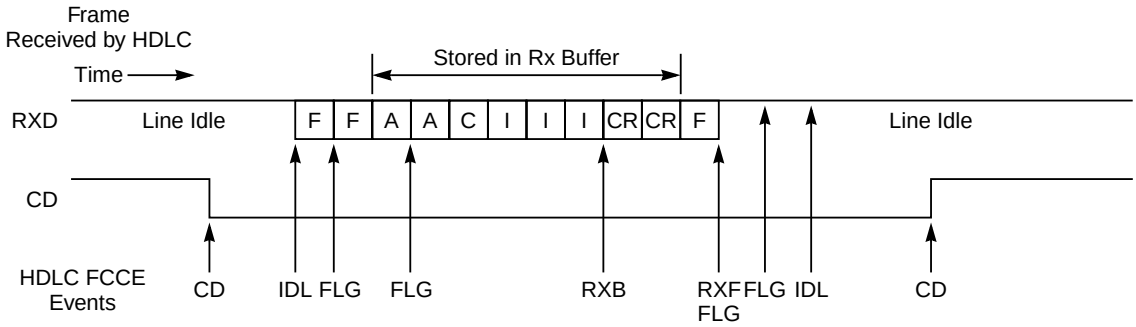
Table 40-9. FCCE/FCCM Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared.
8	GRA	Graceful stop complete. A graceful stop, which was initiated by the GRACEFUL STOP TRANSMIT command, is now complete. GRA is set as soon as the transmitter finishes transmitting any frame that is in progress when the command was issued. It is set immediately if no frame is in progress when the command is issued.
9–10	—	Reserved, should be cleared.
11	TXE	Tx error. An error ($\overline{\text{CTS}}$ lost or underrun) occurs on the transmitter channel.
12	RXF	Rx frame. A complete frame is received on the HDLC channel. This bit is set no sooner than two clocks after receipt of the last bit of the closing flag.
13	BSY	Busy condition. A frame is received and discarded due to a lack of buffers.

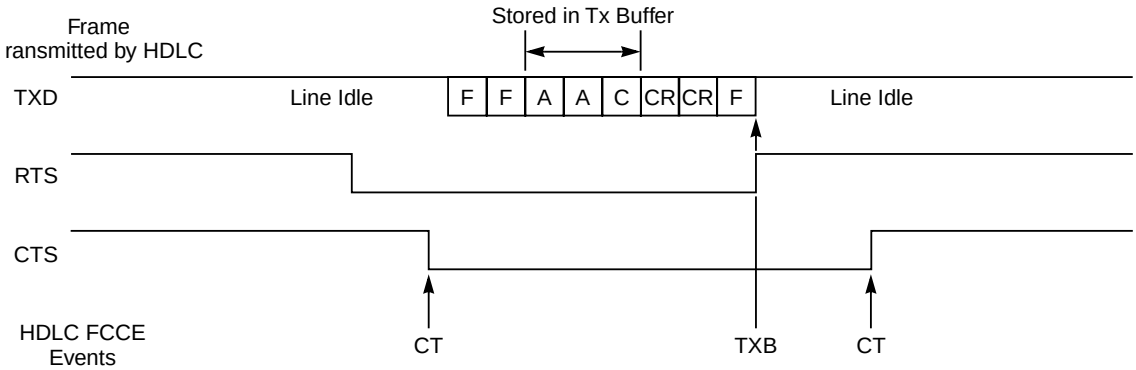
Table 40-9. FCCE/FCCM Field Descriptions (continued)

Bits	Name	Description
14	TXB	Transmit buffer. Enabled by setting TxBD[I]. A buffer is sent on the HDLC channel. TXB is set no sooner than when the last bit of the closing flag begins its transmission if the buffer is the last one in the frame. Otherwise, TXB is set after the last byte of the buffer is written to the transmit FIFO buffer.
15	RXB	Receive buffer. Enabled by setting RxBD[I]. RXB is set when a buffer is filled, even if the buffer is the last in a frame (unlike RXB in the SCCE/SCCM).
16–21	—	Reserved, should be cleared.
22	FLG	Flag status changed. The HDLC controller stops or starts receiving HDLC flags. The real-time status can be obtained in FCCS; see Section 40.10, “FCC Status Register (FCCS).”
23	IDL	Idle sequence status changed. A change in the status of the serial line is detected on the HDLC line. The real-time status can be read in FCCS; see Section 40.10, “FCC Status Register (FCCS).”
24–31	—	Reserved, should be cleared.

Figure 40-8 shows interrupts that can be generated in the HDLC protocol.



Notes:
1. RXB event assumes receive buffers are 6 bytes each.
2. The second IDL event occurs after 15 ones are received in a row.
3. The FLG interrupts show the beginning and end of flag reception.
4. The FLG interrupt at the end of the frame may precede the RXF interrupt due to receive FIFO latency.
5. The CD event must be programmed in the parallel I/O port, not in the FCC itself.
6. F = flag, A = address byte, C = control byte, I = information byte, and CR = CRC byte



Notes:
1. TXB event shown assumes all three bytes were put into a single buffer.
2. Example shows one additional opening flag. This is programmable.
3. The CT event must be programmed in the parallel I/O port, not in the FCC itself.

Figure 40-8. HDLC Interrupt Event Example

40.10 FCC Status Register (FCCS)

The FCCS register, shown in Figure 40-9, allows the user to monitor real-time status conditions on the RXD line. The real-time status of the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ signals are part of the parallel I/O port; see Chapter 42, “Parallel I/O Ports.”

	0–4	5	6	7
Field	—	FG	—	ID
Reset	0000_0000			
R/W	R			
Addr	0x9_1318 (FCCS1), 0x9_1338 (FCCS2), 0x9_1358 (FCCS3)			

Figure 40-9. FCC Status Register (FCCS)

Table 40-10 describes FCCS bits.

Table 40-10. FCCS Register Field Descriptions

Bits	Name	Description
0–4	—	Reserved, should be cleared.
5	FG	Flags. While FG is cleared, each time a new bit is received the most recently received 8 bits are examined to see if a flag is present. FG is set as soon as an HDLC flag (0x7E) is received on the line. Once FG is set, it remains set at least 8 bit times while the next 8 bits of input data are examined. If another flag occurs, FG stays set for at least another eight bits. Otherwise, FG is cleared and the search begins again. 0 HDLC flags are not currently being received. 1 HDLC flags are currently being received.
6	—	Reserved, should be cleared.
7	ID	Idle status. ID is set when the RXD signal is a logic one for 15 or more consecutive bit times; it is cleared after a logic zero is received. 0 The line is busy. 1 The line is idle.

Chapter 41
FCC Transparent Controller

The FCC transparent controller functions as a high-speed serial-to-parallel and parallel-to-serial converter. Transparent mode provides a clear channel on which the FCC performs no bit-level manipulation—implementing higher-level protocols would require software. Transparent mode is also referred to as a totally transparent or promiscuous operation.

Basic applications for an FCC in transparent mode include the following:

- For data, such as voice, moving serially without the need for protocol processing
- For board-level applications, such as chip-to-chip communications, requiring a serial-to-parallel and parallel-to-serial conversion
- For applications requiring the switching of data paths without altering the protocol encoding itself, such as a multiplexer in which data from a high-speed TDM serial stream is divided into multiple low-speed data streams

An FCC transmitter and receiver can be programmed in transparent mode independently. Setting GFMR*x*[TTx] enables the transparent transmitter; setting GFMR*x*[TRx] enables the transparent receiver. Both bits must be set for full-duplex transparent operation. If only one bit is set, the other half of the FCC operates with the protocol programmed in GFMR*x*[MODE]. This allows loopback modes to transfer data from one memory location to another (using DMA) while the data is converted to a specific serial format. However, the Ethernet and ATM controllers cannot be split in this way. See Section 34.2, “General FCC Mode Registers (GFMRx).”

The FCC in transparent mode can work with the TSA or NMSI and support modem lines using the general-purpose I/O signals. The data can be transmitted and received with msb or lsb first in each octet. The FCC consists of separate transmit and receive sections whose operations are asynchronous with the core and can either be synchronous or asynchronous with respect to the other FCCs. Each clock can be supplied from the internal BRG bank or external signals.

41.1 Features

The following is a list of the transparent controller’s important features:

- Flexible data buffers
- Automatic SYNC detection on receive
  - 16-bit pattern
  - 8-bit pattern
  - Automatic sync (always synchronized)
  - External sync signal support
- CRCs can optionally be transmitted and received
- Reverse data mode
- Another protocol can be performed on the FCC's other half (transmitter or receiver) during transparent mode
- External BD table

41.2 Transparent Channel Operation

The transparent transmitter and receiver operates in the same way as the HDLC controller of the FCC (see Chapter 38, "FCC HDLC Controller") except in the following ways:

1. The FPSMR does not affect the transparent controller, only the GFMR does.
2. In Table 38-2., MFLR, HMASK, RFTHR, and RFCNT must be cleared for proper operation of the transparent receiver.
3. Transmitter synchronization has to be achieved using $\overline{\text{CTS}}$ before the transmitter begins sending; see Section 41.3, "Achieving Synchronization in Transparent Mode."

41.3 Achieving Synchronization in Transparent Mode

Once the FCC transmitter is enabled for transparent operation in the GFMR, the TxBD is prepared for the FCC, and the transmit FIFO is preloaded by the SDMA channel, transmit synchronization must be established before data can be sent.

Similarly, once the FCC receiver is enabled for transparent operation in the GFMR and the RxBD is made empty for the FCC, receive synchronization must occur before data can be received. The synchronization process gives the user bit-level control of when the transmission and reception begins. The methods for this are as follows:

- An in-line synchronization pattern
- External synchronization signals
- Automatic sync

41.3.1 In-Line Synchronization Pattern

The transparent channel can be programmed to transmit and receive a synchronization pattern if GFMR[SYNL] ≠ 0; see Section 34.2, “General FCC Mode Registers (GFMRx).” The pattern is defined in the FDSR; see Section 34.5, “FCC Data Synchronization Registers (FDSRx).” GFMR[SYNL] defines the SYNC pattern length. The synchronization pattern is shown in Figure 41-1.

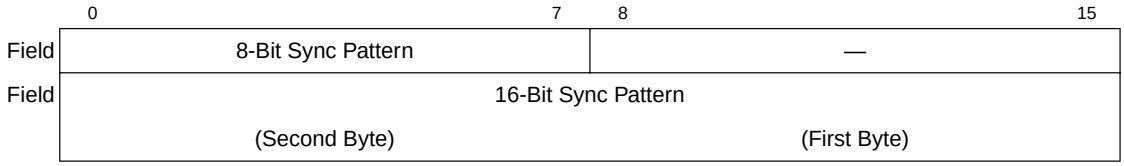


Figure 41-1. In-Line Synchronization Pattern

The receiver synchronizes on the synchronization pattern located in the FDSR. For instance, if an 8-bit SYNC is selected, reception begins as soon as these eight bits are received, beginning with the first bit following the 8-bit SYNC. This effectively links the transmitter synchronization to the receiver synchronization.

41.3.2 External Synchronization Signals

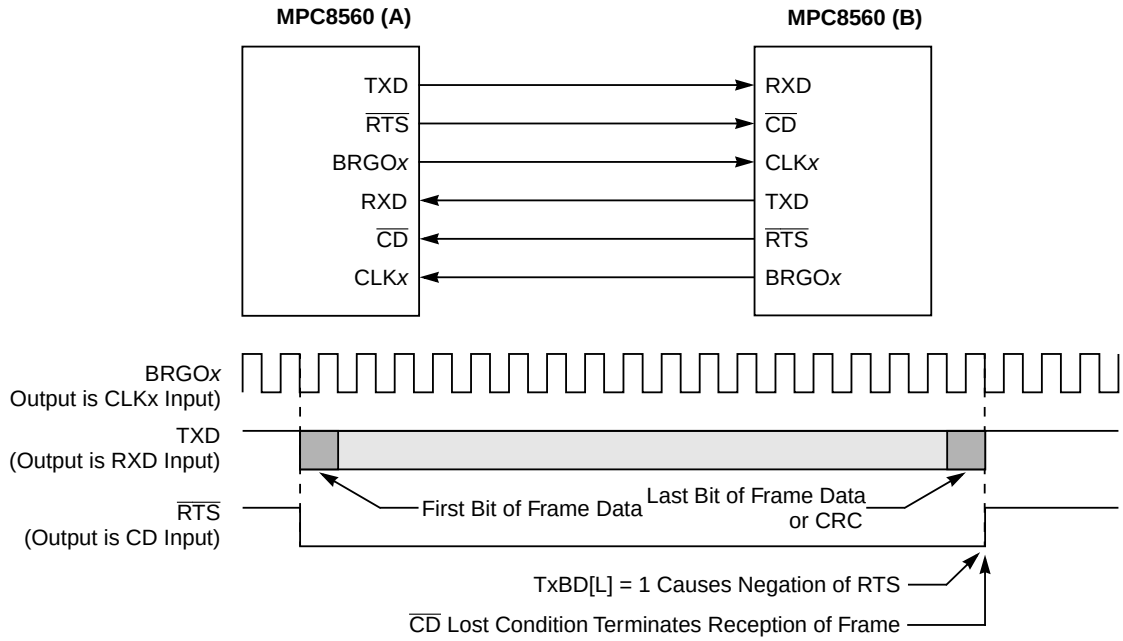
If GFMR[SYNL] = 00, an external signal is used to begin the sequence. $\overline{\text{CTS}}$ is used for the transmitter and $\overline{\text{CD}}$ is used for the receiver; these signals share the following sampling options.

- The pulse/envelope option determines whether $\overline{\text{CD}}$ or $\overline{\text{CTS}}$ need to be asserted only once to begin reception/transmission or whether they must be asserted and stay that way for the duration of the transparent frame. This option is controlled by the CDP and CTSP bits of the GFMR. If the user expects a continuous stream of data without interruption, the pulse option should be used. However, if the user needs to identify frames of transparent data, the envelope mode of the these signals should be used. Note that the first bit of a frame is transmitted as zero every time $\overline{\text{RTS}}$ is asserted before $\overline{\text{CTS}}$ is asserted (GFMR[CTSS] = 1); subsequent data bits are sent accurately. Similarly, if $\overline{\text{CTS}}$ is in pulse mode (GFMR[CTSP] = 1), only the first frame is affected. If $\overline{\text{CTS}}$ is not in pulse mode (GFMR[CTSP] = 0), every frame is affected separately.
- The sampling option determines the delay between $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ being asserted and the resulting action by the FCC. These signals can be assumed to be asynchronous to the data and then internally synchronized by the FCC, or they can be assumed to be synchronous to the data giving faster operation. This option allows the $\overline{\text{RTS}}$ of one FCC to be connected to the $\overline{\text{CD}}$ of another FCC (on another MPC8560) and to have the data synchronized and bit aligned. It is also an option to link the transmitter synchronization to the receiver synchronization.

When working with the FCC receiver in envelope mode, $\overline{\text{RTS}}$ should be asserted for at least 3 clock cycles between frames. Otherwise, the receiver cannot recognize the start of a new frame. Diagrams for the pulse/envelope and sampling options are in Section 34.12, “FCC Timing Control.”

41.3.3 Transparent Synchronization Example

Figure 41-2 shows an example of synchronization using external signals.



Notes:
- 1. Each MPC8560 generates its own transmit clocks. If the transmit and receive clocks are the same, one can generate transmit and receive clocks for the other MPC8560. For example, CLK*x* on MPC8560 (B) could be used to clock the transmitter and receiver.
- 2. $\overline{\text{CTS}}$ should be configured as always asserted in the parallel I/O port or connected to ground externally.
- 3. The required GSMR configurations are DIAG= 00, CTSS=1, CTSP is a don’t care, CDS=1, CDP=0, TTX=1, and TRX=1. REVD and TCRC are application-dependent.
- 4. The transparent frame contains a CRC if TxBD[TC] is set.

Figure 41-2. Sending Transparent Frames Between MPC8560s

MPC8560(A) and MPC8560(B) exchange transparent frames and synchronize each other using $\overline{\text{RTS}}$ and $\overline{\text{CD}}$. However, $\overline{\text{CTS}}$ is not required because transmission begins at any time. Thus, $\overline{\text{RTS}}$ is connected directly to the other MPC8560’s $\overline{\text{CD}}$. GFMR[SYNL] is not used and transmission and reception from each MPC8560 are independent.

Chapter 42
Serial Peripheral Interface (SPI)

The serial peripheral interface (SPI) allows the MPC8560 to exchange data between other MPC8560 devices, the MPC8260, the MPC860, the MC68360, the MC68302, the M68HC11 and M68HC05 microcontroller families, and peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices.

The SPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (receive, transmit, clock and slave select). The SPI block consists of transmitter and receiver sections, an independent baud-rate generator, and a control unit. The transmitter and receiver sections use the same clock, which is derived from the SPI baud rate generator in master mode and generated externally in slave mode. During an SPI transfer, data is sent and received simultaneously.

Because the SPI receiver and transmitter are double-buffered, as shown in Figure 42-1, the effective FIFO size (latency) is 2 characters. The SPI's msb is shifted out first. When the SPI is disabled in the SPI mode register (SPMODE[EN] = 0), it consumes little power.

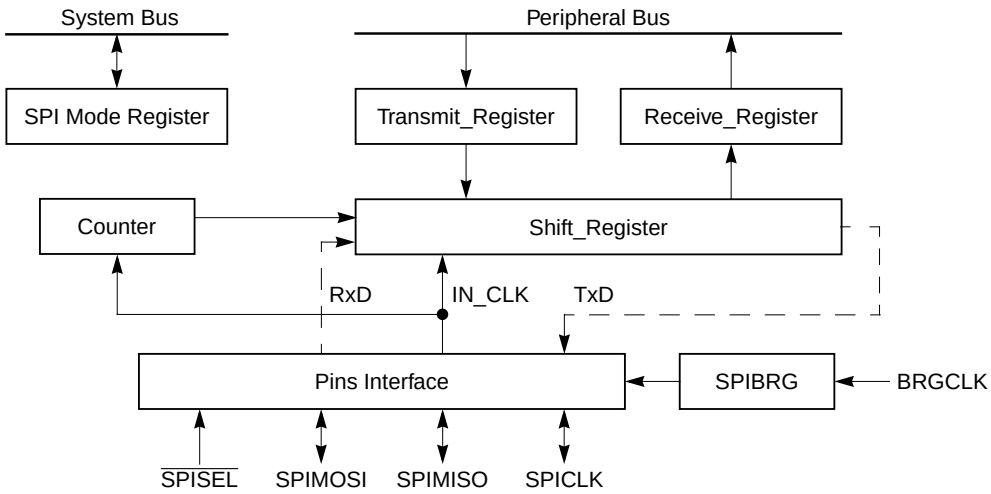


Figure 42-1. SPI Block Diagram

42.1 Features

The following is a list of the SPI's main features:

- Four-signal interface (SPIMOSI, SPIMISO, SPICLK, and $\overline{\text{SPISEL}}$) multiplexed with port D signals
- Full-duplex operation
- Works with data characters from 4 to 16 bits long
- Supports back-to-back character transmission and reception
- Master or slave SPI modes supported
- Multimaster environment support
- Continuous transfer mode for automatic scanning of a peripheral
- Supports maximum clock rates of 25 in master mode and 50 MHz in slave mode, assuming a 100-MHz system clock
- Independent programmable baud rate generator
- Programmable clock phase and polarity
- Open-drain outputs support multimaster configuration
- Local loopback capability for testing

42.2 SPI Clocking and Signal Functions

The SPI can be configured as a slave or as a master in single- or multiple-master environments. The master SPI generates the transfer clock SPICLK using the SPI baud rate generator (BRG). The SPI BRG takes its input from BRGCLK, which is generated in the MPC8560 clock synthesizer.

SPICLK is a gated clock, active only during data transfers. Four combinations of SPICLK phase and polarity can be configured with SPMODE[CI, CP]. SPI signals can also be configured as open-drain to support a multimaster configuration in which a shared SPI signal is driven by the MPC8560 or an external SPI device.

The SPI master-in slave-out SPIMISO signal acts as an input for master devices and as an output for slave devices. Conversely, the master-out slave-in SPIMOSI signal is an output for master devices and an input for slave devices. The dual functionality of these signals allows the SPIs in a multimaster environment to communicate with one another using a common hardware configuration.

- When the SPI is a master, SPICLK is the clock output signal that shifts received data in from SPIMISO and transmitted data out to SPIMOSI. SPI masters must output a slave select signal to enable SPI slave devices by using a separate general-purpose I/O signal. Assertion of an SPI's $\overline{\text{SPISEL}}$ while it is master causes an error.
- When the SPI is a slave, SPICLK is the clock input that shifts received data in from SPIMOSI and transmitted data out through SPIMISO. $\overline{\text{SPISEL}}$ is the enable input to the SPI slave. In a multimaster environment, $\overline{\text{SPISEL}}$ (always an input) is used to detect an error when more than one master is operating.

As described in Chapter 42, “Parallel I/O Ports,” SPIMISO, SPIMOSI, SPICLK, and $\overline{\text{SPISEL}}$ are multiplexed with port D[16–19] signals, respectively. They are configured as SPI signals through the port D signal assignment register (PDPAR) and the port D data direction register (PDDIR), specifically by setting PDPAR[DD*n*] and PDDIR[DR*n*].

42.3 Configuring the SPI Controller

The SPI can be programmed to work in a single- or multiple-master environment. This section describes SPI master and slave operation in a single-master configuration and then discusses the multi-master environment.

42.3.1 The SPI as a Master Device

In master mode, the SPI sends a message to the slave peripheral, which sends back a simultaneous reply. A single master MPC8560 with multiple slaves can use general-purpose parallel I/O signals to selectively enable slaves, as shown in Figure 42-2. To eliminate the multimaster error in a single-master environment, the master’s $\overline{\text{SPISEL}}$ input can be forced inactive by selecting port D[19] for general-purpose I/O (PDPAR[DD19] = 0).

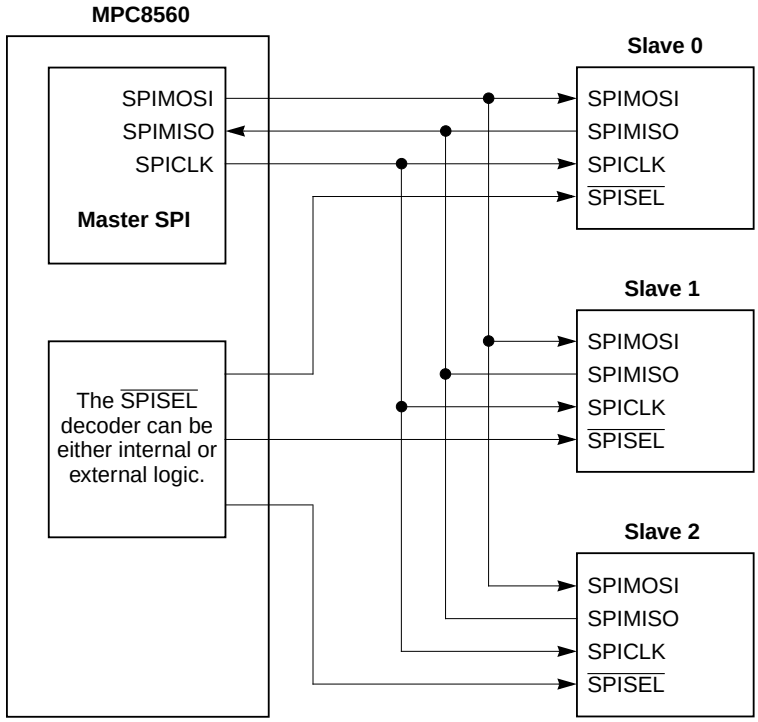


Figure 42-2. Single-Master/Multi-Slave Configuration

To start exchanging data, the core writes the data to be sent into a buffer, configures a TxBD with TxBD[R] set, and configures one or more RxBDs. The core then sets SPCOM[STR] in the SPI command register to start sending data, which starts once the SDMA channel loads the Tx FIFO with data.

The SPI then generates programmable clock pulses on SPICLK for each character and simultaneously shifts Tx data out on SPIMOSI and Rx data in on SPIMISO. Received data is written into a Rx buffer using the next available RxBD. The SPI keeps sending and receiving characters until the whole buffer is sent or an error occurs. The CP then clears TxBD[R] and RxBD[E] and issues a maskable interrupt to the interrupt controller in the SIU.

When multiple TxBDs are ready, TxBD[L] determines whether the SPI keeps transmitting without SPCOM[STR] being set again. If the current TxBD[L] is cleared, the next TxBD is processed after data from the current buffer is sent. Typically there is no delay on SPIMOSI between buffers. If the current TxBD[L] is set, sending stops after the current buffer is sent. In addition, the RxBD is closed after transmission stops, even if the Rx buffer is not full; therefore, Rx buffers need not be the same length as Tx buffers.

42.3.2 The SPI as a Slave Device

In slave mode, the SPI receives messages from an SPI master and sends a simultaneous reply. The slave's $\overline{\text{SPISEL}}$ must be asserted before Rx clocks are recognized; once $\overline{\text{SPISEL}}$ is asserted, SPICLK becomes an input from the master to the slave. SPICLK can be any frequency from DC to BRGCLK/2 (12.5 MHz for a 25-MHz system).

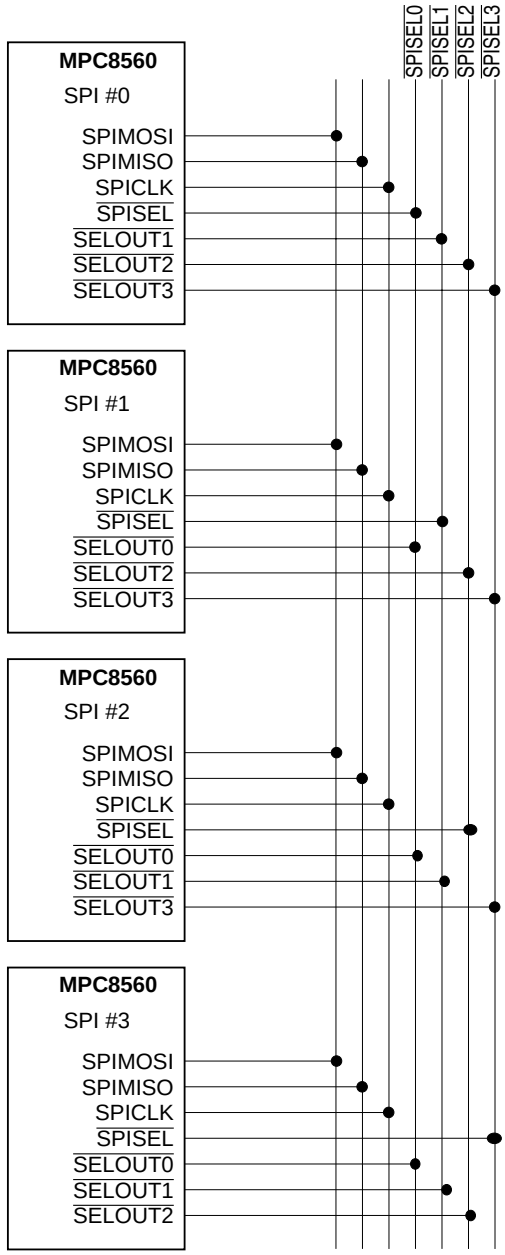
To prepare for data transfers, the slave's core writes data to be sent into a buffer, configures a TxBD with TxBD[R] set, and configures one or more RxBDs. The core then sets SPCOM[STR] to activate the SPI. Once $\overline{\text{SPISEL}}$ is asserted, the slave shifts data out from SPIMISO and in through SPIMOSI. A maskable interrupt is issued when a full buffer finishes receiving and sending or after an error. The SPI uses successive RxBDs in the table to continue reception until it runs out of Rx buffers or $\overline{\text{SPISEL}}$ is negated.

Transmission continues until no more data is available or $\overline{\text{SPISEL}}$ is negated. If it is negated before all data is sent, it stops but the TxBD stays open. Transmission continues once $\overline{\text{SPISEL}}$ is reasserted and SPICLK begins toggling. After the characters in the buffer are sent, the SPI sends ones as long as $\overline{\text{SPISEL}}$ remains asserted.

42.3.3 The SPI in Multimaster Operation

The SPI can operate in a multimaster environment in which SPI devices are connected to the same bus. In this configuration, the SPIMOSI, SPIMISO, and SPICLK signals of all SPIs are shared; the $\overline{\text{SPISEL}}$ inputs are connected separately, as shown in Figure 42-3. Only one SPI device can act as master at a time—all others must be slaves. When an SPI is configured as a master and its $\overline{\text{SPISEL}}$ input is asserted, a multimaster error occurs because

more than one SPI device is a bus master. The SPI sets SPIE[MME] in the SPI event register and a maskable interrupt is issued to the core. It also disables SPI operation and the output drivers of SPI signals. The core must clear SPMODE[EN] before the SPI is used again. After correcting the problems, clear SPIE[MME] and reenable the SPI.



Notes:
- All signals are open-drain
- For a system with more than two masters, SPISEL and SPIE[MME] do not detect all possible conflicts
- It is the responsibility of software to arbitrate for the SPI bus (with token passing, for example)
- SELOUT*x* signals are implemented in software with general-purpose I/O signals

Figure 42-3. Multimaster Configuration

The maximum sustained data rate that the SPI supports is SYSTEMCLK/50. However, the SPI can transfer a single character at much higher rates—SYSTEMCLK/4 in master mode and SYSTEMCLK/2 in slave mode. Gaps should be inserted between multiple characters to keep from exceeding the maximum sustained data rate.

42.4 Programming the SPI Registers

The following sections describe the registers used in configuring and operating the SPI.

42.4.1 SPI Mode Register (SPMODE)

The SPI mode register (SPMODE), shown in Figure 42-4, controls both the SPI operation mode and clock source.

	0	1	2	3	4	5	6	7	8 – 11	12 – 15
Field	—	**LOOP**	**CI**	**CP**	**DIV16**	**REV**	**M/S**	**EN**	**LEN**	**PM**
Reset	0000_00						—	0_0000_0000		
R/W	R/W									
Addr	0x9_1AA0									

Figure 42-4. SPMODE—SPI Mode Register

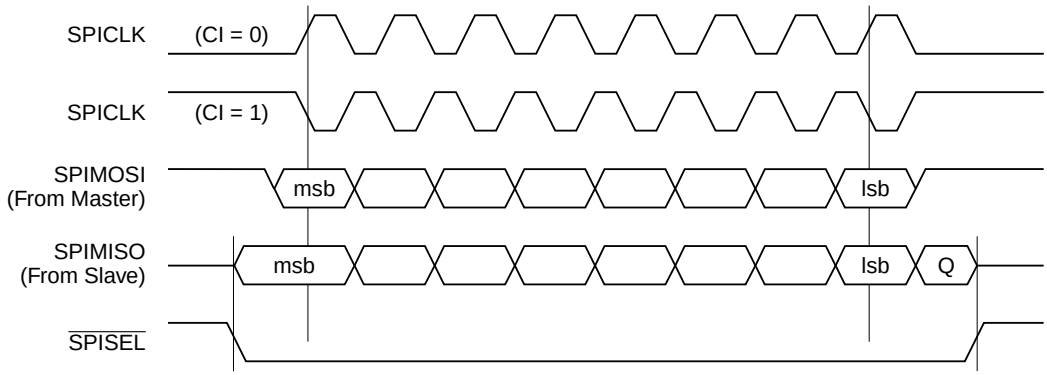
Table 42-1 describes the SPMODE fields.

Table 42-1. SPMODE Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared.
1	**LOOP**	Loop mode. Enables local loopback operation. 0 Normal operation. 1 Loopback mode. The transmitter output is internally connected to the receiver input. The receiver and transmitter operate normally, except that received data is ignored.
2	**CI**	Clock invert. Inverts SPI clock polarity. See Figure 42-5 and Figure 42-6. 0 The inactive state of SPICLK is low. 1 The inactive state of SPICLK is high.
3	**CP**	Clock phase. Selects the transfer format. See Figure 42-5 and Figure 42-6. 0 SPICLK starts toggling at the middle of the data transfer. 1 SPICLK starts toggling at the beginning of the data transfer.
4	**DIV16**	Divide by 16. Selects the clock source for the SPI baud rate generator when configured as an SPI master. In slave mode, SPICLK is the clock source. 0 BRGCLK is the input to the SPI BRG. 1 BRGCLK/16 is the input to the SPI BRG.
5	**REV**	Reverse data. Determines the receive and transmit character bit order. 0 Reverse data—lsb of the character sent and received first. 1 Normal operation—msb of the character sent and received first.

Table 42-1. SPMODE Field Descriptions (continued)

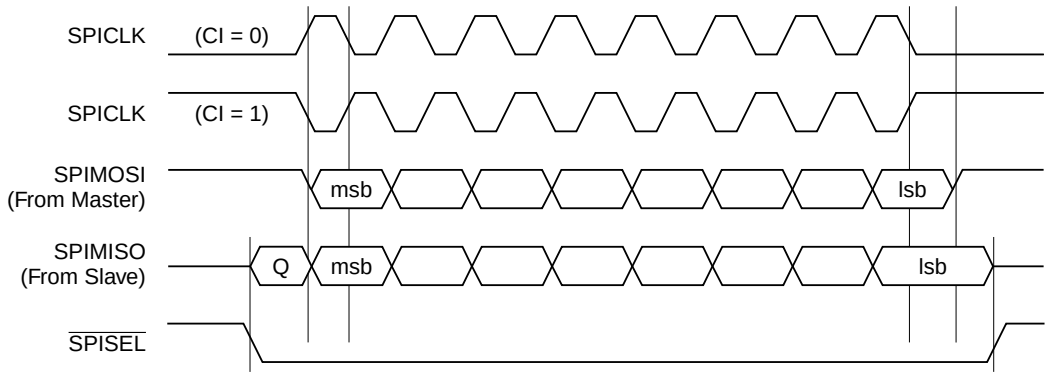
Bits	Name	Description
6	**M/S**	Master/slave. Selects master or slave mode. 0 The SPI is a slave. 1 The SPI is a master.
7	**EN**	Enable SPI. Do not change other SPMODE bits when EN is set. 0 The SPI is disabled. The SPI is in a reset state and consumes minimal power. The SPI BRG is not functioning and the input clock is disabled. 1 The SPI is enabled. Configure SPIMOSI, SPIMISO, SPICLK, and $\overline{\text{SPISEL}}$ to connect to the SPI as described in Section 42.2, “Port Registers. “
8–11	**LEN**	Character length in bits per character. If the character length is not greater than a byte, every byte in memory holds (LEN+1) valid bits. If the character length is greater than a byte, every half-word holds (LEN+1) valid bits. See Section 42.4.1.1, “SPI Examples with Different SPMODE[LEN] Values.” 0000–0010 Reserved, causes erratic behavior. 0011 4-bit characters … 1111 16-bit characters
12–15	**PM**	Prescale modulus select. Specifies the divide ratio of the prescale divider in the SPI clock generator. BRGCLK is divided by 4 * ([PM0–PM3] + 1), a range from 4 to 64. The clock has a 50% duty cycle.



NOTE: Q = Undefined Signal.

Figure 42-5. SPI Transfer Format with SPMODE[CP] = 0

Figure 42-6 shows the SPI transfer format in which SPICLK starts toggling at the beginning of the transfer (SPMODE[CP] = 1).



NOTE: Q = Undefined Signal.

Figure 42-6. SPI Transfer Format with SPMODE[CP] = 1

42.4.1.1 SPI Examples with Different SPMODE[LEN] Values

The examples below show how SPMODE[LEN] is used to determine character length. To help map the process, the conventions shown in Table 42-2 are used in the examples.

Table 42-2. Example Conventions

Convention	Description
g–v	Binary symbols
x	Deleted bit
__ [1]	Original byte boundary
_ [1]	Original 4-bit boundary.

[1] Both __ and _ are used to aid readability.

Once the data string image is determined, it is always transmitted byte by byte with the lsb of the most-significant byte sent first. For all examples below, assume the memory contains the following binary image:

```
             msb         ghij_klmn__opqr_stuv             lsb
```

Example 1

```
with LEN=4 (data size=5), the following data is selected:
             msb         xxxj_klmn__xxxr_stuv             lsb
with REV=0, the data string image is:
             msb         j_klmn__r_stuv                   lsb
the order of the string appearing on the line, a byte at a time is:
             first       nmlk_j__vuts_r                   last

with REV=1,the string has each byte reversed, and the data string image is:
             msb         nmlk_j__vuts_r                   lsb
the order of the string appearing on the line, one byte at a time is:
```

```
              first       j_klmn__r_stuv                    last
```

Example 2

```
with LEN=7 (data size=8), the following data is selected:
              msb         ghij_klmn__opqr_stuv              lsb
the data string is selected:
              msb         ghij_klmn__opqr_stuv              lsb
with REV=0, the string transmitted, a byte at a time with lsb first is:
              first       nmlk_jihg__vuts_rqpo              last

with REV=1, the string is byte reversed and transmitted, a byte at a time, with
lsb first:
              first       ghij_klmn__opqr_stuv              last
```

Example 3:

```
with LEN=0xC (data size=13), the following data is selected:
              msb         ghij_klmn__xxxr_stuv              lsb
the data string selected is:
              msb         r_stuv__ghij_klmn                 lsb
with REV=0, the string transmitted, a byte at a time with lsb first is:
              first       vuts_r__nmlk_jihg                 last

with REV=1, the string is half-word reversed:
              msb         nmlk_jihg__vuts_r                 lsb
and transmitted a byte at a time with lsb first:
              first       ghij_klmn__r_stuv                 last
```

42.4.2 SPI Event/Mask Registers (SPIE/SPIM)

The SPI event register (SPIE) generates interrupts and reports events recognized by the SPI. When an event is recognized, the SPI sets the corresponding SPIE bit. Clear SPIE bits by writing a 1—writing 0 has no effect. Setting a bit in the SPI mask register (SPIM) enables and clearing a bit masks the corresponding interrupt. Unmasked SPIE bits must be cleared before the CP clears internal interrupt requests. Figure 42-7 shows both registers.

	0	1	2	3	4	5	6	7
Field	—		MME	TXE	—	BSY	TXB	RXB
Reset	0000_0000							
R/W	R/W							
Addr	0x9_1AA6 (SPIE); 0x9_1AAA (SPIM)							

Figure 42-7. SPIE/SPIM—SPI Event/Mask Registers

Table 42-3 describes the SPIE/SPIM fields.

Table 42-3. SPIE/SPIM Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	MME	Multimaster error. Set when $\overline{\text{SPISEL}}$ is asserted externally while the SPI is in master mode.
3	TXE	Tx error. Set when an error occurs during transmission.
4	—	Reserved, should be cleared.
5	BSY	Busy. Set after the first character is received but discarded because no Rx buffer is available.
6	TXB	Tx buffer. Set when the Tx data of the last character in the buffer is written to the Tx FIFO. Wait two character times to be sure data is completely sent over the transmit signal.
7	RXB	Rx buffer. Set after the last character is written to the Rx buffer and the BD is closed.

42.4.3 SPI Command Register (SPCOM)

The SPI command register (SPCOM), shown in Figure 42-8, is used to start SPI operation.

	0	1 … 7
Field	**STR**	—
Reset	0000_0000	
R/W	Write Only	
Addr	0x9_1AAD	

Figure 42-8. SPCOM—SPI Command Register

Table 42-4 describes the SPCOM fields.

Table 42-4. SPCOM Field Descriptions

Bits	Name	Description
0	**STR**	Start transmit. For an SPI master, setting STR causes the SPI to start transferring data to and from the Tx/Rx buffers if they are prepared. For a slave, setting STR when the SPI is idle causes it to load the Tx data register from the SPI Tx buffer and start sending with the next SPICLK after $\overline{\text{SPISEL}}$ is asserted. STR is cleared automatically after one system clock cycle.
1–7	—	Reserved and should be cleared.

42.5 SPI Parameter RAM

The SPI parameter RAM area is similar to the SCC general-purpose parameter RAM. The CP accesses the SPI parameter table using a user-programmed pointer (SPI_BASE) located in the parameter RAM; see Section 20.5.2, “Parameter RAM.” The SPI parameter table can be placed at any 64-byte aligned address in the dual-port RAM’s general-purpose area (banks 1–8, 11 and 12). Some parameter values must be user-initialized before the SPI is enabled; the CP initializes the others. Once initialized, parameter RAM values do not

usually need to be accessed. They should be changed only when the SPI is inactive. Table 42-5 shows the memory map of the SPI parameter RAM.

Table 42-5. SPI Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x00	**RBASE**	Hword	Rx/Tx BD table base address. Indicate where the BD tables begin in the dual-port RAM. Setting Rx/TxBD[W] in the last BD in each BD table determines how many BDs are allocated for the Tx and Rx sections of the SPI. Initialize RBASE/TBASE before enabling the SPI. Furthermore, do not configure BD tables of the SPI to overlap any other active controller's parameter RAM. RBASE and TBASE should be divisible by eight.
0x02	**TBASE**	Hword	
0x04	**RFCR**	Byte	Rx/Tx function code registers. The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. See Section 42.5.1, "Receive/Transmit Function Code Registers (RFCR/TFCR)."
0x05	**TFCR**	Byte	
0x06	**MRBLR**	Hword	Maximum receive buffer length. The SPI has one MRBLR entry to define the maximum number of bytes the MPC8560 writes to a Rx buffer before moving to the next buffer. The MPC8560 can write fewer bytes than MRBLR if an error or end-of-frame occurs, but never exceeds the MRBLR value. User-supplied buffers should be no smaller than MRBLR. Tx buffers are unaffected by MRBLR and can have varying lengths; the number of bytes to be sent is programmed in TxBD[Data Length]. MRBLR is not intended to be changed while the SPI is operating. However it can be changed in a single bus cycle with one 16-bit move (not two 8-bit bus cycles back-to-back). The change takes effect when the CP moves control to the next RxBD. To guarantee the exact RxBD on which the change occurs, change MRBLR only while the SPI receiver is disabled. MRBLR should be greater than zero; it should be an even number if the character length of the data exceeds 8 bits.
0x08	RSTATE	Word	Rx internal state. [2] Reserved for CP use.
0x0C	—	Word	The Rx internal data pointer [2] is updated by the SDMA channels to show the next address in the buffer to be accessed.
0x10	RBPTR	Hword	RxBD pointer. Points to the current Rx BD being processed or to the next BD to be serviced when idle. After a reset or when the end of the BD table is reached, the CP initializes RBPTR to the RBASE value. Most applications should not modify RBPTR, but it can be updated when the receiver is disabled or when no Rx buffer is in use.
0x12	—	Hword	The Rx internal byte count [2] is a down-count value that is initialized with the MRBLR value and decremented with every byte the SDMA channels write.
0x14	—	Word	Rx temp.[2] Reserved for CP use.
0x18	TSTATE	Word	Tx internal state.[2] Reserved for CP use.
0x1C	—	Word	The Tx internal data pointer[2] is updated by the SDMA channels to show the next address in the buffer to be accessed.
0x20	TBPTR	Hword	TxBD pointer. Points to the current Tx BD during frame transmission or the next BD to be processed when idle. After reset or when the end of the Tx BD table is reached, the CP initializes TBPTR to the TBASE value. Most applications do not need to modify TBPTR, but it can be updated when the transmitter is disabled or when no Tx buffer is in use.
0x22	—	Hword	The Tx internal byte count[2] is a down-count value initialized with TxBD[Data Length] and decremented with every byte read by the SDMA channels.
0x24	—	Word	Tx temp.[2] Reserved for CP use.
0x34	—	Word	SDMA temp.

1 From the pointer value programmed in SPI_BASE at IMMR + 0x89FC.

2 Normally, these parameters need not be accessed. They are listed to help experienced users in debugging.

42.5.1 Receive/Transmit Function Code Registers (RFCR/TFCR)

Figure 42-9 shows the fields in the receive/transmit function code registers (RFCR/TFCR).

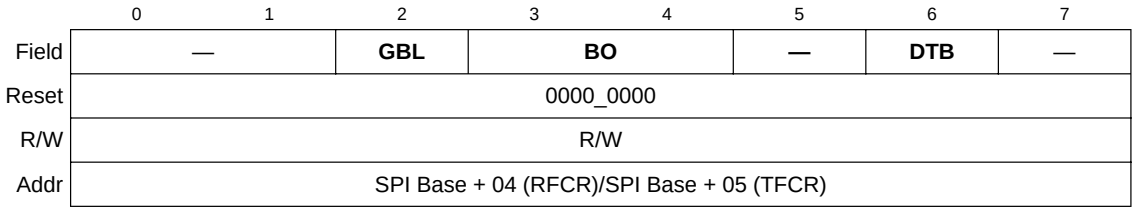


Figure 42-9. RFCR/TFCR—Function Code Registers

Table 42-6 describes the RFCR/TFCR fields.

Table 42-6. RFCR/TFCR Field Descriptions

Bits	Name	Description
0–1	**—**	Reserved, should be cleared.
2	**GBL**	Global access bit 0 Disable memory snooping 1 Enable memory snooping
3–4	**BO**	Byte ordering. Set BO to select the required byte ordering for the buffer. If BO is changed on the fly, it takes effect at the beginning of the next frame or BD. 00 Little-endian 01 Big-endian 01 Reserved 11 Reserved
5	**—**	Reserved, should be cleared.
6	**DTB**	Data bus indicator. 0 Use system bus for SDMA operation. 1 Use local bus for SDMA operation.
7	**—**	Reserved, should be cleared.

42.6 SPI Commands

Table 42-7 lists transmit/receive commands sent to the CP command register (CPCR).

Table 42-7. SPI Commands

Command	Description
INIT TX PARAMETERS	Initializes all transmit parameters in the parameter RAM to their reset state and should be issued only when the transmitter is disabled. The INIT TX AND RX PARAMETERS command can also be used to reset both the Tx and Rx parameters.
CLOSE RXBD	Forces the SPI controller to close the current RxBD and use the next BD for subsequently received data. If the controller is not receiving data, no action is taken. Use this command to extract data from a partially full buffer.
INIT RX PARAMETERS	Initializes all receive parameters in the parameter RAM to their reset state. Should be issued only when the receiver is disabled. The INIT TX AND RX PARAMETERS command can also be used to reset both the Tx and Rx parameters.

42.7 The SPI Buffer Descriptor (BD) Table

As shown in Figure 42-10, BDs are organized into separate RxBD and TxBD tables in dual-port RAM. The tables have the same basic configuration as for the SCCs and form circular queues that determine the order buffers are transferred. The CP uses BDs to confirm reception and transmission or to indicate error conditions so that the core knows buffers have been serviced. The buffers themselves can be placed in external memory or in any unused parameter area of the dual-port RAM.

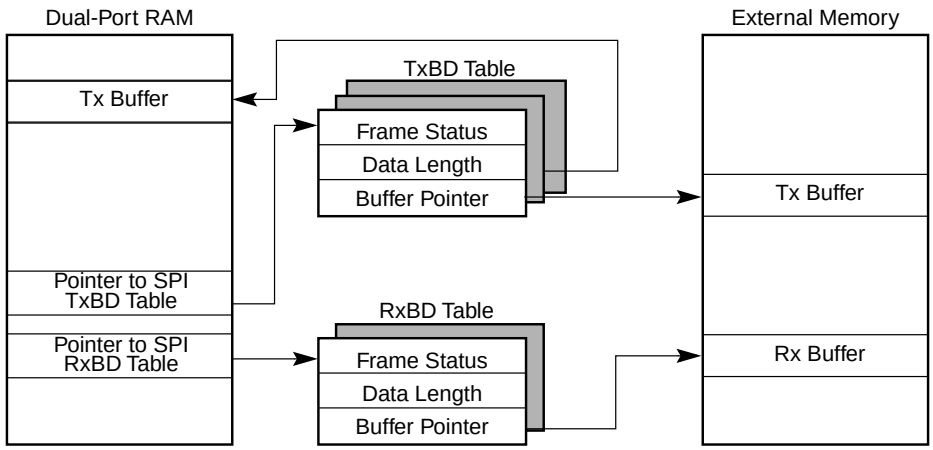


Figure 42-10. SPI Memory Structure

42.7.1 SPI Buffer Descriptors (BDs)

Receive and transmit BDs report information about each buffer transferred and whether a maskable interrupt should be generated. Each 64-bit BD, shown in Figure 42-11 and Figure 42-12, has the following structure:

- The half word at offset + 0 contains status and control bits. The CP updates the status bits after the buffer is sent or received.

- The half word at offset + 2 contains the data length (in bytes) that is sent or received.
  - For an RxBD, this is the number of octets the CP writes into this RxBD's buffer once the BD closes. The CP updates this field after the received data is placed into the buffer. Memory allocated for this buffer should be no smaller than MRBLR.
  - For a TxBD, this is the number of octets the CP should transmit from its buffer. Normally, this value should be greater than zero. If the character length is more than 8 bits, the data length should be even. For example, to send three characters of 8-bit data, 1 start, and 1 stop, the data length field should be initialized to 3. However, to send three characters of 9-bit data, the data length field should be initialized to 6 since the three 9-bit data fields occupy three half-words in memory. The CP never modifies this field.
- The word at offset + 4 points to the beginning of the buffer.
  - For an RxBD, the pointer must be even and can point to internal or external memory.
  - For a TxBD, the pointer can be even or odd, unless the character exceeds 8 bits, for which it must be even. The buffer can be in internal or external memory.

42.7.1.1 SPI Receive BD (RxBD)

The CP uses RxBDs to report on each received buffer. It closes the current buffer, generates a maskable interrupt, and starts receiving data in the next buffer once the current buffer is full. The CP also closes the buffer when the SPI is configured as a slave and $\overline{\text{SPISEL}}$ is negated, indicating that reception stopped. The core should write RxBD bits before the SPI is enabled. The format of an RxBD is shown in Figure 42-11.

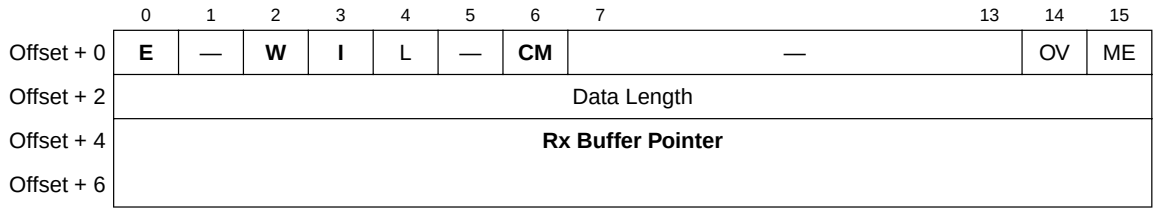


Figure 42-11. SPI RxBD

Table 42-8 describes the RxBD status and control fields.

Table 42-8. SPI RxBD Status and Control Field Descriptions

Bits	Name	Description
0	**E**	Empty. 0 The buffer is full or stopped receiving because of an error. The core can examine or write to any fields of this RxBD, but the CP does not use this BD while E = 0. 1 The buffer is empty or reception is in progress. The CP owns this RxBD and its buffer. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved, should be cleared.
2	**W**	Wrap (last BD in table). 0 Not the last BD in the RxBD table. 1 Last BD in the RxBD table. After this buffer is used, the CP receives incoming data using the BD pointed to by RBASE (top of the table). The number of BDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is filled. 1 SPIE[RXB] is set when this buffer is full, indicating the need for the core to process the buffer. SPIE[RXB] causes an interrupt if not masked.
4	L	Last. Updated by the SPI when the buffer is closed because $\overline{\text{SPISEL}}$ was negated (slave mode only). Otherwise, RxBD[ME] is set. The SPI updates L after received data is placed in the buffer. 0 This buffer does not contain the last character of the message. 1 This buffer contains the last character of the message.
5	—	Reserved, should be cleared.
6	**CM**	Continuous mode. Master mode only; in slave mode, CM should be cleared. 0 Normal operation. 1 The CP does not clear RxBD[E] after this BD is closed; the buffer is overwritten when the CP next accesses this BD. This allows continuous reception from an SPI slave into one buffer for autoscanning of a serial A/D peripheral with no core overhead.
7–13	—	Reserved, should be cleared.
14	OV	Overrun. Set when a receiver overrun occurs during reception (slave mode only). The SPI updates OV after the received data is placed in the buffer.
15	ME	Multimaster error. Set when this buffer is closed because $\overline{\text{SPISEL}}$ was asserted when the SPI was in master mode. Indicates a synchronization problem between multiple masters on the SPI bus. The SPI updates ME after the received data is placed in the buffer.

42.7.1.2 SPI Transmit BD (TxBD)

Data to be sent with the SPI is sent to the CP by arranging it in buffers referenced by TxBDs in the TxBD table. TxBD fields should be prepared before data is sent. The format of an TxBD is shown in Figure 42-12.

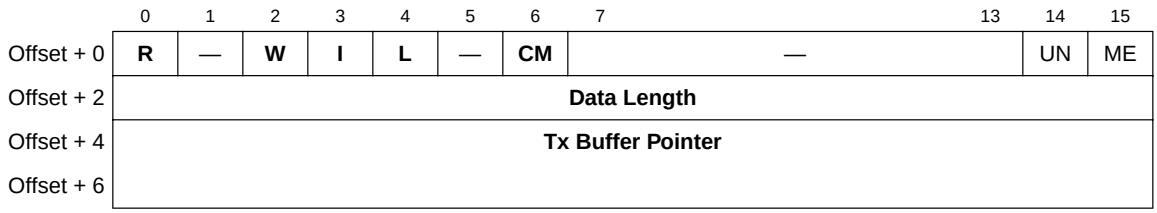


Figure 42-12. SPI TxBD

Table 42-9 describes the TxBD status and control fields.

Table 42-9. SPI TxBD Status and Control Field Descriptions

Bits	Name	Description
0	**R**	Ready. 0 The buffer is not ready to be sent. This BD or its buffer can be modified. The CP clears R (unless RxBD[CM] is set) after the buffer is sent (unless RxBD[CM] is set) or an error occurs. 1 The buffer is ready for transmission or is being sent. The BD cannot be modified once R is set.
1	—	Reserved, should be cleared.
2	**W**	Wrap (last BD in TxBD table). 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CP receives incoming data using the BD pointed to by TBASE (top of the table). The number of BDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	**I**	Interrupt. 0 No interrupt is generated after this buffer is processed. 1 SPIE[TXB] or SPIE[TXE] are set when this buffer is processed and causes interrupts if not masked.
4	**L**	Last. 0 This buffer does not contain the last character of the message. 1 This buffer contains the last character of the message.
5	—	Reserved, should be cleared.
6	**CM**	Continuous mode. Valid only when the SPI is in master mode. In slave mode, it should be cleared. 0 Normal operation. 1 The CP does not clear TxBD[R] after this BD is closed, allowing the buffer to be resent automatically when the CP next accesses this BD.
7–13	—	Reserved, should be cleared.
14	UN	Underrun. Indicates that the SPI encountered a transmitter underrun condition while sending the buffer. This error occurs only when the SPI is in slave mode. The SPI updates UN after it sends the buffer.
15	ME	Multimaster error. Indicates that this buffer is closed because $\overline{\text{SPISEL}}$ was asserted when the SPI was in master mode. A synchronization problem occurred between devices on the SPI bus. The SPI updates ME after sending the buffer.

42.8 SPI Master Programming Example

The following sequence initializes the SPI to run at a high speed in master mode:

1. Configure port D to enable SPIMISO, SPIMOSI, SPICLK and $\overline{\text{SPISEL}}$.
2. Configure a parallel I/O signal to operate as the SPI select output signal if needed.

3. In address 0x89FC, assign a pointer to the SPI parameter RAM.
4. Write RBASE and TBASE in the SPI parameter RAM to point to the RxBD and TxBD tables in the dual-port RAM. Assuming one RxBD followed by one TxBD at the beginning of the dual-port RAM, write RBASE with 0x0000 and TBASE with 0x0008.
5. Write RFCR and TFCR with 0x10 for normal operation.
6. Write MRBLR with the maximum number of bytes per Rx buffer. For this case, assume 16 bytes, so MRBLR = 0x0010.
7. Initialize the RxBD. Assume the Rx buffer is at 0x0000_1000 in main memory. Write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x0000_1000 to RxBD[Buffer Pointer].
8. Initialize the TxBD. Assume the Tx buffer is at 0x0000_2000 in main memory and contains five 8-bit characters. Write 0xB800 to TxBD[Status and Control], 0x0005 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer].
9. Execute the INIT RX AND TX PARAMETERS command by writing 0x2541_0000 to CPCR.
10. Write 0xFF to SPIE to clear any previous events.
11. Write 0x37 to SPIM to enable all possible SPI interrupts.
12. Write 0x0370 to SPMODE to enable normal operation (not loopback), master mode, SPI enabled, 8-bit characters, and the fastest speed possible.
13. Set SPCOM[STR] to start the transfer.

After 5 bytes are sent, the TxBD is closed. Additionally, the Rx buffer is closed after 5 bytes are received because TxBD[L] is set.

42.9 SPI Slave Programming Example

The following is an example initialization sequence to follow when the SPI is in slave mode. It is very similar to the SPI master example, except that $\overline{\text{SPISEL}}$ is used instead of a general-purpose I/O signal (as shown in Figure 42-2).

1. Enable SPIMISO, SPIMOSI, SPICLK, and $\overline{\text{SPISEL}}$.
2. In address 0x89FC, assign a pointer to the SPI parameter RAM.
3. Assuming one RxBD at the beginning of the dual-port RAM followed by one TxBD, write RBASE with 0x0000 and TBASE with 0x0008 in the SPI parameter RAM.
4. Write RFCR and TFCR with 0x10 for normal operation.
5. Program MRBLR = 0x0010 for 16 bytes, the maximum number of bytes per buffer.

6. Initialize the RxBD. Assume the Rx buffer is at 0x0000_1000 in main memory. Write 0xB000 to RxBD[Status and Control], 0x0000 to RxBD[Data Length] (optional), and 0x0000_1000 to RxBD[Buffer Pointer].
7. Initialize the TxBD. Assume the Tx buffer is at 0x0000_2000 in main memory and contains five 8-bit characters. Write 0xB800 to TxBD[Status and Control], 0x0005 to TxBD[Data Length], and 0x0000_2000 to TxBD[Buffer Pointer].
8. Execute the INIT RX AND TX PARAMETERS command by writing 0x2541_0000 to CPCR.
9. Write 0xFF to SPIE to clear any previous events.
10. Write 0x37 to SPIM to enable all SPI interrupts.
11. Set SPMODE to 0x0170 to enable normal operation (not loopback), slave mode, SPI enabled, and 8-bit characters. BRG speed is ignored in slave mode.
12. Set SPCOM[STR] to enable the SPI to be ready once the master begins the transfer.

Note that if the master sends 3 bytes and negates $\overline{\text{SPISEL}}$, the RxBD is closed but the TxBD remains open. If the master sends 5 or more bytes, the TxBD is closed after the fifth byte. If the master sends 16 bytes and negates $\overline{\text{SPISEL}}$, the RxBD is closed without triggering an out-of-buffers error. If the master sends more than 16 bytes, the RxBD is closed (full) and an out-of-buffers error occurs after the 17th byte is received.

42.10 Handling Interrupts in the SPI

The following sequence should be followed to handle interrupts in the SPI:

1. Once an interrupt occurs, read SPIE to determine the interrupt source. Normally, SPIE bits should be cleared at this time.
2. Process the TxBD to reuse it and the RxBD to extract the data from it. To transmit another buffer, simply set TxBD[R], RxBD[E], and SPCOM[STR].
3. Execute an **rfi** instruction.

Chapter 43
CPM I²C Controller

The CPM inter-integrated circuit (I²C®) controller lets the MPC8560 exchange data with other I²C devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCD displays through the CPM. The CPM I²C controller uses a synchronous, multimaster bus that can connect several integrated circuits on a board. It uses two signals—serial data (SDA) and serial clock (SCL)—to carry information between the integrated circuits connected to it.

As shown in Figure 32-1, this I²C controller consists of transmit and receive sections, an independent baud-rate generator (BRG), and a control unit. The transmit and receive sections use the same clock, which is derived from the I²C BRG when in master mode and generated externally when in slave mode. Wait states are inserted during a data transfer if SCL is held low by a slave device. In the middle of a data transfer, the master I²C controller recognizes the need for wait states by monitoring SCL. However, the I²C controller has no automatic time-out mechanism if the slave device does not release SCL; therefore, software should monitor how long SCL stays low to generate bus timeouts.

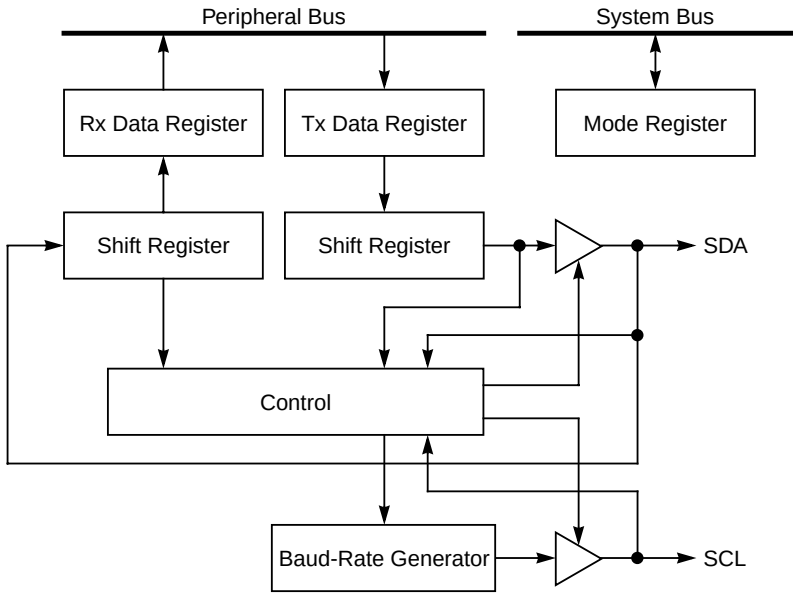


Figure 43-1. I²C Controller Block Diagram

The I^2C receiver and transmitter are double-buffered, which corresponds to an effective two-character FIFO latency. In normal operation, the transmitter shifts the msb (bit 0) out first. When the I^2C is not enabled in the I^2C mode register (I2MOD[EN] = 0), it consumes little power.

43.1 Features

The following is a list of the I^2C controller's main features:

- Two-signal interface (SDA and SCL)
- Support for master and slave I^2C operation
- Multiple-master environment support
- Continuous transfer mode for automatic scanning of a peripheral
- Supports a maximum clock rate of 2,080 KHz (with a CPM utilization of 25%), assuming a 100-MHz system clock.
- Independent, programmable baud-rate generator
- Supports 7-bit I^2C addressing
- Open-drain output signals allow multiple master configuration
- Local loopback capability for testing

43.2 I^2C Controller Clocking and Signal Functions

The I^2C controller can be configured as a master or slave for the serial channel. As a master, the controller's BRG provides the transfer clock. The I^2C BRG takes its input from the BRG clock (BRGCLK), which is generated from the CPM clock.

SDA and SCL are bidirectional signals connected to a positive supply voltage through an external pull-up resistor. When the bus is free, both signals are pulled high. The general I^2C master/slave configuration is shown in Figure 32-2.

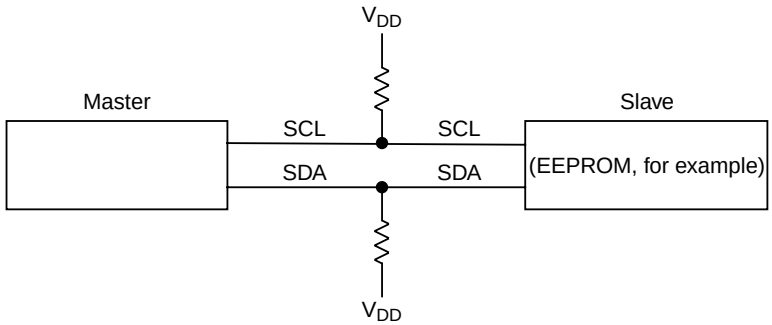


Figure 43-2. I^2C Master/Slave General Configuration

When the I²C controller is master, the SCL clock output, taken directly from the I²C BRG, shifts receive data in and transmit data out through SDA. The transmitter arbitrates for the bus during transmission and aborts if it loses arbitration. When the I²C controller is a slave, the SCL clock input shifts data in and out through SDA. The SCL frequency can range from DC to BRGCLK/48.

43.3 I²C Controller Transfers

To initiate a transfer, the master I²C controller sends a message specifying a read or write request to an I²C slave. The first byte of the message consists of a 7-bit slave port address and a R/W request bit. Note that because the R/W request follows the slave port address in the I²C bus specification, the R/W request bit must be placed in the lsb (bit 7) unless operating in reverse data mode; see Section 43.4.1, "I²C Mode Register (I2MOD)."

To write to a slave, the master sends a write request (R/W = 0) along with either the target slave's address or a general call (broadcast) address of all zeros, followed by the data to be written. To read from a slave, the master sends a read request (R/W = 1) and the target slave's address. When the target slave acknowledges the read request, the transfer direction is reversed, and the master receives the slave's transmit buffer(s). If the receiver (master or slave) does not acknowledge each byte transfer in the ninth bit frame, the transmitter signals a transmission error event (I2ER[TXE]). An I²C transfer timing diagram is shown in Figure 32-3.

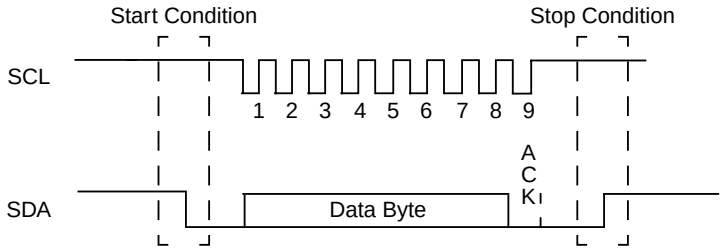


Figure 43-3. I²C Transfer Timing

Select master or slave mode for the controller using the I²C command register (I2COM[M/S]). Set the master's start bit, I2COM[STR], to begin a transfer; setting a slave's I2COM[STR] activates the slave to wait for a transfer request from a master.

If a master or slave transmitter's current TxBD[L] is set, transmission stops once the buffer is sent; that is, I2COM[STR] must be set again to reactivate transfers. If TxBD[L] is zero, once the current buffer is sent, the controller begins processing the next TxBD without waiting for I2COM[STR] to be set again.

The following sections further detail the transfer process.

43.3.1 I2C Master Write (Slave Read)

If the MPC8560 is the master, prepare the transmit buffers and BDs before initiating a write. Initialize the first transmit data byte with the slave address and write request (R/W = 0).

If the MPC8560 is the slave target of the write, prepare receive buffers and BDs to await the master's request. Figure 32-4 shows the timing for a master write.

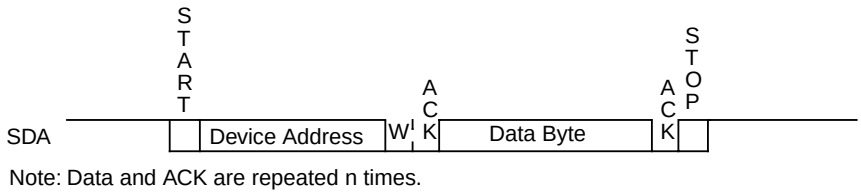


Figure 43-4. I2C Master Write Timing

A master write occurs as follows:

1. The master core sets I2COM[STR]. The transfer starts when the SDMA channel loads the Tx FIFO with data and the I2C bus is not busy.
2. The I2C master generates a start condition—a high-to-low transition on SDA while SCL is high—and the transfer clock SCL pulses for each bit shifted out on SDA. If the master transmitter detects a multiple-master collision (by sensing a '0' on SDA while sending a '1'), transmission stops and the channel reverts to slave mode. A maskable interrupt is sent to the master's core so software can try to retransmit later.
3. The slave acknowledges each byte and writes to its current receive buffer until a new start or stop condition is detected.
4. After sending each byte, the master monitors the acknowledge indication. If the slave receiver fails to acknowledge a byte, transmission stops and the master generates a stop condition—a low-to-high transition on SDA while SCL is high.

43.3.2 I2C Loopback Testing

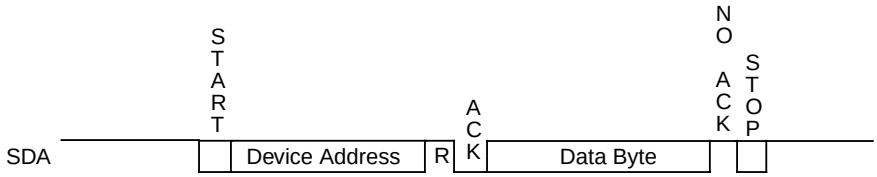
When in master mode, an I2C controller supports loopback operation for master write requests. The master I2C controller simply issues a write request directed to its own address (programmed in I2ADD). The master's receiver monitors the transmission and reads the transmitted data into its receive buffer. Loopback operation requires no special register programming.

43.3.3 I2C Master Read (Slave Write)

Before initiating a master read with the MPC8560, prepare a transmit buffer of size *n*+1 bytes, where *n* is the number of bytes to be read from the slave. The first transmit byte should be initialized to the slave address with R/W = 1. The next *n* transmit bytes are used

strictly for timing and can be left uninitialized. Configure suitable receive buffers and BDs to receive the slave's transmission.

If the MPC8560 is the slave target of the read, prepare the I²C transmit buffers and BDs and activate it by setting I2COM[STR]. Figure 43-5 shows the timing for a master read.



Note: After the nth data byte, the master does not acknowledge the slave.

Figure 43-5. I²C Master Read Timing

A master read occurs as follows:

1. Set the master's I2COM[STR] to initiate the read. The transfer starts when the SDMA channel loads the transmit FIFO with data and the I²C bus is not busy.
2. The slave detects a start condition on SDA and SCL.
3. After the first byte is shifted in, the slave compares the received data to its slave address. If the slave is an MPC8560, the address is programmed in its I²C address register (I2ADD).
   - — If a match is found, the slave acknowledges the received byte and begins transmitting on the clock pulse immediately following the acknowledge.
   - — If a match is found but the slave is not ready, the read request is not acknowledged and the transaction is aborted. If the slave is an MPC8560, a maskable transmission error interrupt is triggered to allow software to prepare data for transmission on the next try.
   - — If a mismatch occurs, the slave ignores the message and searches for a new start condition.
4. The master acknowledges each byte sent as long as an overrun does not occur. If the master receiver fails to acknowledge a byte, the slave aborts transmission. For a slave MPC8560, the abort generates a maskable interrupt. A maskable interrupt is also issued after a complete buffer is sent or after an error. If an underrun occurs, the MPC8560 slave sends ones until a stop condition is detected.

43.3.4 I²C Multi-Master Considerations

The I²C controller supports a multi-master configuration, in which the I²C controller must alternate between master and slave modes. The I²C controller supports this by implementing I²C master arbitration in hardware. However, due to the nature of the I²C bus and the implementation of the I²C controller, certain software considerations must be made.

An MPC8560 I2C controller attempting a master read request could simultaneously be targeted for an external master write (slave read). Both operations trigger the controller's I2CER[RXB] event, but only one operation wins the bus arbitration. To determine which operation caused the interrupt, software must verify that its transmit operation actually completed before assuming that the received data is the result of its read operation.

Problems could also arise if the MPC8560's I2C controller master sets up a transmit buffer and BD for a write request, but then is the target of a read request from another master. Without software precautions, the I2C controller responds to the other master with the transmit buffer originally intended for its own write request. To avoid this situation, a higher-level handshake protocol must be used. For example, a master, before reading a slave, writes the slave with a description of the requested data (which register should be read, for example). This operation is typical with many I2C devices.

43.4 I2C Registers

The following sections describe the I2C registers.

43.4.1 I2C Mode Register (I2MOD)

The I2C mode register, shown in Figure 43-6, controls the I2C modes and clock source.

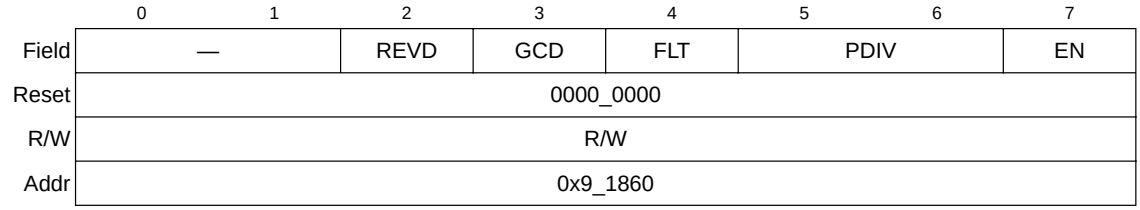


Figure 43-6. I2C Mode Register (I2MOD)

Table 43-1 describes I2MOD bit functions.

Table 43-1. I2MOD Field Descriptions

Bits	Name	Description
0–1	—	Reserved and should be cleared.
2	REVD	Reverse data. Determines the Rx and Tx character bit order. 0 Normal operation. The msb (bit 0) of a character is transferred first. 1 Reverse data. the lsb (bit 7) of a character is transferred first. Note: Clearing REVD is strongly recommended to ensure consistent bit ordering across devices.
3	GCD	General call disable. Determines whether the receiver acknowledges a general call address. 0 General call address is enabled. 1 General call address is disabled.
4	FLT	Clock filter. Determines if the I2C input clock SCL is filtered to prevent spikes in a noisy environment. 0 SCL is not filtered. 1 SCL is filtered by a digital filter.

Table 43-1. I2MOD Field Descriptions (continued)

Bits	Name	Description
5–6	PDIV	Predivider. Selects the clock division factor before it is input into the I²C BRG. The clock source for the I²C BRG is the BRGCLK generated from the CPM clock. 00 BRGCLK/32 01 BRGCLK/16 10 BRGCLK/8 11 BRGCLK/4 Note: To both save power and reduce noise susceptibility, select the PDIV with the largest division factor (slowest clock) that still meets performance requirements.
7	EN	Enable I²C operation. 0 I²C is disabled. TheI²C is in a reset state and consumes minimal power. 1 I²C is enabled. Do not change other I2MOD bits when EN is set.

43.4.2 I²C Address Register (I2ADD)

The I²C address register, shown in Figure 43-7, holds the address for this I²C port.

	0	6	7
Field	SAD		—
Reset	Undefined		
R/W	R/W		
Addr	0x9_1864		

Figure 43-7. I²C Address Register (I2ADD)

Table 43-2 describes I2CADD fields.

Table 43-2. I2ADD Field Descriptions

Bits	Name	Description
0–6	SAD	Slave address 0–6. Holds the slave address for the I²C port.
7	—	Reserved and should be cleared.

43.4.3 I²C Baud Rate Generator Register (I2BRG)

The I²C baud rate generator register, shown in Figure 43-8, sets the divide ratio of the I²C BRG.

	0	7
Field	DIV	
Reset	1111_1111	
R/W	R/W	
Addr	0x9_1868	

Figure 43-8. I²C Baud Rate Generator Register (I2BRG)

Table 43-3 describes I2BRG fields.

Table 43-3. I2BRG Field Descriptions

Bits	Name	Description
0–7	DIV	Division ratio 0–7. Specifies the divide ratio of the BRG divider in the I²C clock generator. The output of the prescaler is divided by 2 × ([DIV0–DIV7] + 3) and the clock has a 50% duty cycle. DIV must be programmed to a minimum value of 2 if the digital filter is disabled and 6 if it is enabled.

43.4.4 I²C Event/Mask Registers (I2CER/I2CMR)

The I²C event register (I2CER) is used to generate interrupts and report events. When an event is recognized, the I²C controller sets the corresponding I2CER bit. I2CER bits are cleared by writing ones; writing zeros has no effect. Setting a bit in the I²C mask register (I2CMR) enables and clearing a bit masks the corresponding interrupt. Unmasked I2CER bits must be cleared before the CP clears internal interrupt requests. Figure 43-9 shows both registers.

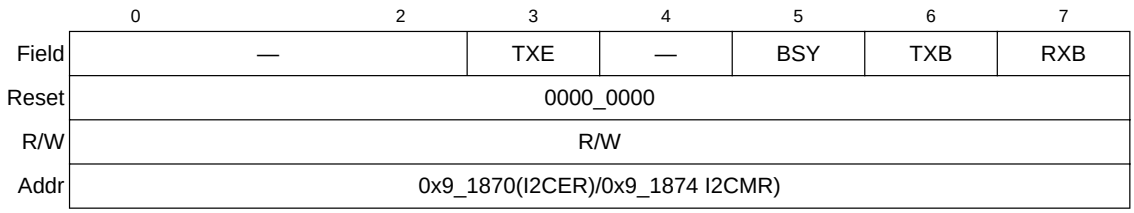


Figure 43-9. I²C Event/Mask Registers (I2CER/I2CMR)

Table 43-4 describes the I2CER/I2CMR fields.

Table 43-4. I2CER/I2CMR Field Descriptions

Bits	Name	Description
0–2	—	Reserved and should be cleared.
3	TXE	Tx error. Set when an error occurs during transmission.
4	—	Reserved and should be cleared.
5	BSY	Busy. Set after the first character is received but discarded because no Rx buffer is available.
6	TXB	Tx buffer. Set when the Tx data of the last character in the buffer is written to the Tx FIFO. Two character times must elapse to guarantee that all data has been sent.
7	RXB	Rx buffer. Set after the last character is written to the Rx buffer and the RxBD is closed.

43.4.5 I²C Command Register (I2COM)

The I²C command register, shown in Figure 43-10, is used to start I²C transfers and to select master or slave mode.

	0	1 – 6	7
Field	STR	—	M/S
Reset	0000_0000		
R/W	R/W		
Addr	0x9_186C		

Figure 43-10. I2C Command Register (I2COM)

Table 43-5 describes I2COM fields.

Table 43-5. I2COM Field Descriptions

Bits	Name	Description
0	STR	Start transmit. In master mode, setting STR causes the I2C controller to start sending data from the I2C Tx buffers if they are ready. In slave mode, setting STR when the I2C controller is idle causes it to load the Tx data register from the I2C Tx buffer and start sending when it receives an address byte that matches the slave address with R/W = 1. STR is always read as a 0.
1–6	—	Reserved and should be cleared.
7	M/S	Master/slave. Configures the I2C controller to operate as a master or a slave. 0 I2C is a slave. 1 I2C is a master.

43.5 I2C Parameter RAM

The I2C controller parameter table is used for the general I2C parameters and is similar to the SCC general-purpose parameter RAM. The CP accesses the I2C parameter table using a user-programmed pointer (I2C_BASE) located in the parameter RAM; see Section 20.5.2, “Parameter RAM.” The I2C parameter table can be placed at any 64-byte aligned address in the dual-port RAM’s general-purpose area (banks 1–8, 11 and 12). The user must initialize certain parameter RAM values before the I2C is enabled; the CP initializes the other values. Software usually does not access parameter RAM entries once they are initialized; they should be changed only when the I2C is inactive.

Table 43-6 describes the I2C parameter RAM memory map.

Table 43-6. I2C Parameter RAM Memory Map

Offset [1]	Name	Width	Description
0x00	RBASE	Hword	Rx/TxBD table base address. Indicate where the BD tables begin in the dual-port RAM. Setting Rx/TxBD[W] in the last BD in each BD table determines how many BDs are allocated for the Tx and Rx sections of the I2C. Initialize RBASE/TBASE before enabling the I2C. Furthermore, do not configure BD tables of the I2C to overlap any other active controller’s parameter RAM. RBASE and TBASE should be divisible by eight.
0x02	TBASE	Hword	
0x04	RFCR	Byte	Rx/Tx function code registers. The function code registers contain the transaction specification associated with SDMA channel accesses to external memory. See Figure 43-11 and Table 43-7.
0x05	TFCR	Byte	

Table 43-6. I²C Parameter RAM Memory Map (continued)

Offset [1]	Name	Width	Description
0x06	MRBLR	Hword	Maximum receive buffer length. Defines the maximum number of bytes the MPC8560 writes to a Rx buffer before moving to the next buffer. The MPC8560 writes fewer bytes to the buffer than the MRBLR value if an error or end-of-frame occurs. Buffers should not be smaller than MRBLR. Tx buffers are unaffected by MRBLR and can vary in length; the number of bytes to be sent is specified in TxBD[Data Length]. MRBLR is not intended to be changed while the I²C is operating. However it can be changed in a single bus cycle with one 16-bit move (not two 8-bit bus cycles back-to-back). The change takes effect when the CP moves control to the next RxBD. To guarantee the exact RxBD on which the change occurs, change MRBLR only while the I²C receiver is disabled. MRBLR should be greater than zero; it should be an even number if the character length of the data exceeds 8 bits.
0x08	RSTATE	Word	Rx internal state. [2] Reserved for CP use.
0x0C	RPTR	Word	Rx internal data pointer [2] is updated by the SDMA channels to show the next address in the buffer to be accessed.
0x10	RBPTR	Hword	RxBD pointer. Points to the next descriptor the receiver transfers data to when it is in an idle state or to the current descriptor during frame processing for each I²C channel. After a reset or when the end of the descriptor table is reached, the CP initializes RBPTR to the value in RBASE. Most applications should not write RBPTR, but it can be modified when the receiver is disabled or when no receive buffer is used.
0x12	RCOUNT	Hword	Rx internal byte count [2] is a down-count value that is initialized with the MRBLR value and decremented with every byte the SDMA channels write.
0x14	RTEMP	Word	Rx temp. [2] Reserved for CP use.
0x18	TSTATE	Word	Tx internal state. [2] Reserved for CP use.
0x1C	TPTR	Word	Tx internal data pointer [2] is updated by the SDMA channels to show the next address in the buffer to be accessed.
0x20	TBPTR	Hword	TxBD pointer. Points to the next descriptor that the transmitter transfers data from when it is in an idle state or to the current descriptor during frame transmission. After a reset or when the end of the descriptor table is reached, the CP initializes TBPTR to the value in TBASE.Most applications should not write TBPTR, but it can be modified when the transmitter is disabled or when no transmit buffer is used.
0x22	TCOUNT	Hword	Tx internal byte count [2] is a down-count value initialized with TxBD[Data Length] and decremented with every byte read by the SDMA channels.
0x24	TTEMP	Word	Tx temp. [2] Reserved for CP use.
0x34	SDMATMP	Word	SDMA temp. [2] Reserved for CP use.

1 From the pointer value programmed in I2C_BASE at IMMR + 0x8AFC.

2 Normally, these parameters need not be accessed.

Figure 43-11 shows the RFCR/TFCR bit fields.

	0	1	2	3	4	5	6	7
Field			GBL	BO		—	DTB	—
Reset	0000_0000							
R/W	R/W							
Addr	I2C_BASE + 04 (RFCR)/I2C_BASE + 05 (TFCR)							

Figure 43-11. I2C Function Code Registers (RFCR/TFCR)

Table 43-7 describes the RFCR/TFCR bit fields.

Table 43-7. RFCR/TFCR Field Descriptions

Bits	Name	Description
0–1	—	Reserved, should be cleared.
2	GBL	Global access bit 0 Disable memory snooping 0 Enable memory snooping
3–4	BO	Byte ordering. Selects the required byte ordering for the buffer. If BO is changed on-the-fly, it takes effect at the beginning of the next frame or BD. 0x Reserved. 1x Big-endian.
5	—	Reserved, should be cleared.
6	DTB	Data bus indicator. 0 Use system bus for SDMA operation. 1 Use local bus for SDMA operation.
7	—	Reserved, should be cleared.

43.6 I2C Commands

The I2C transmit and receive commands, shown in Table 43-8, are issued to the CP command register (CPCR).

Table 43-8. I2C Transmit/Receive Commands

Command	Description
INIT TX PARAMETERS	Initializes all transmit parameters in the parameter RAM to their reset state. Should be issued only when the transmitter is disabled. The INIT TX AND RX PARAMETERS command can also be used to reset both the Tx and Rx parameters.
CLOSE RXBD	Forces the I2C controller to close the current Rx BD and use the next BD for subsequently received data. If the controller is not receiving data, no action is taken. Use this command to extract data from a partially full buffer.
INIT RX PARAMETERS	Initializes all receive parameters in the parameter RAM to their reset state. Should be issued only when the receiver is disabled. The INIT TX AND RX PARAMETERS command can also be used to reset both the Tx and Rx parameters.

43.7 I2C Buffer Descriptor (BD) Tables

As shown in Figure 43-12, buffer descriptors (BDs) are organized into separate RxBD and TxBD tables in dual-port RAM. The tables have the same basic configuration as for the SCCs and form circular queues that determine the order buffers are transferred. The CP uses BDs to confirm reception and transmission or to indicate error conditions so that the core knows buffers have been serviced. The buffers themselves can be placed in external memory or in any unused parameter area of the dual-port RAM.

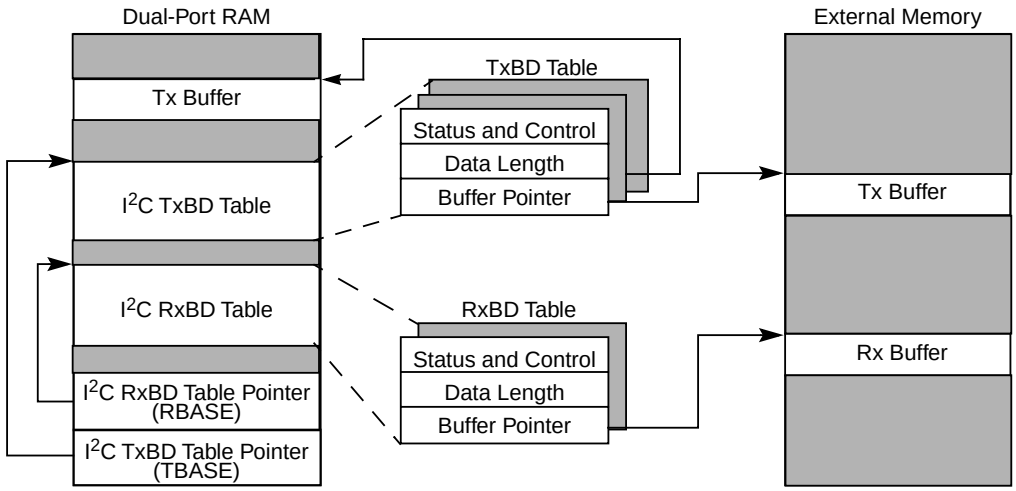


Figure 43-12. I2C Memory Structure

43.8 I2C Buffer Descriptors (BDs)

Receive and transmit buffer descriptors report information about each buffer transferred and whether a maskable interrupt should be generated. Each 64-bit BD, shown in Figure 43-13 and Figure 43-14, has the following structure:

- The half word at offset + 0 contains status and control bits. The CP updates the status bits after the buffer is sent or received.
- The half word at offset + 2 contains the data length (in bytes) that is sent or received.
- For an RxBD, this is the number of octets the CP writes into this RxBD's buffer once the descriptor closes. The CP updates this field after the received data is placed into the associated buffer. Memory allocated for this buffer should be no smaller than MRBLR.
- For a TxBD, this is the number of octets the CP should transmit from its buffer. Normally, this value should be greater than zero. The CP never modifies this field.
- The word at offset + 4 points to the beginning of the buffer.
- For an RxBD, the pointer must be even and can point to internal or external memory.

- For a TxBD, the pointer can be even or odd. The buffer can reside in internal or external memory.

43.8.0.1 I²C Receive Buffer Descriptor (RxBD)

Using RxBDs, the CP reports on each buffer received, closes the current buffer, generates a maskable interrupt, and starts receiving data in the next buffer when the current one is full. It closes the buffer when a stop or start condition is found on the I²C bus or when an overrun error occurs. The core should write RxBD bits before the I²C controller is enabled.

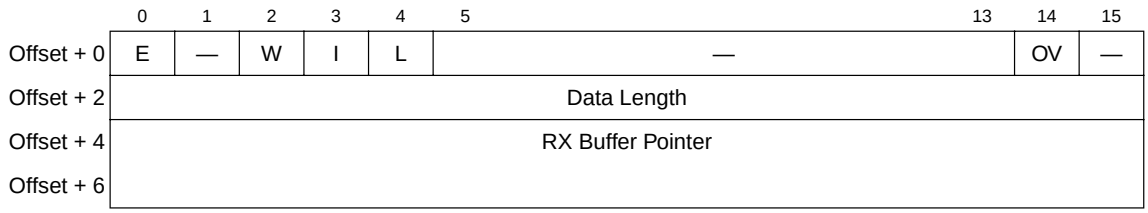


Figure 43-13. I²C RxBD

Table 43-9 describes I²C RxBD status and control bits.

Table 43-9. I²C RxBD Status and Control Bits

Bits	Name	Description
0	E	Empty. 0 The buffer is full or stopped receiving because of an error. The core can examine or write to any fields of this RxBD, but the CP does not use this BD while E = 0. 1 The buffer is empty or reception is in progress. The CP owns this RxBD and its buffer. Once E is set, the core should not write any fields of this RxBD.
1	—	Reserved and should be cleared.
2	W	Wrap (last BD in table). 0 Not the last BD in the RxBD table. 1 Last BD in the RxBD table. After this buffer is used, the CP receives incoming data using the BD pointed to by RBASE (top of the table). The number of BDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	I	Interrupt. 0 No interrupt is generated after this buffer is full. 1 The I2CER[RXB] is set when the CP fills this buffer, indicating that the core needs to process the buffer. The RXB bit can cause an interrupt if it is enabled.
4	L	Last. The I²C controller sets L. 0 This buffer does not contain the last character of the message. 1 This buffer holds the last character of the message. The I²C controller sets L after all received data is placed into the associated buffer, or because of a stop or start condition or an overrun.
5–13	—	Reserved and should be cleared.
14	OV	Overrun. Set when a receiver overrun occurs during reception. The I²C controller updates this bit after the received data is placed into the associated buffer.
15	—	Reserved and should be cleared.

43.8.0.2 I2C Transmit Buffer Descriptor (TxBD)

Transmit data is arranged in buffers referenced by TxBDs in the TxBD table. The first word of the TxBD, shown in Figure 43-14, contains status and control bits.

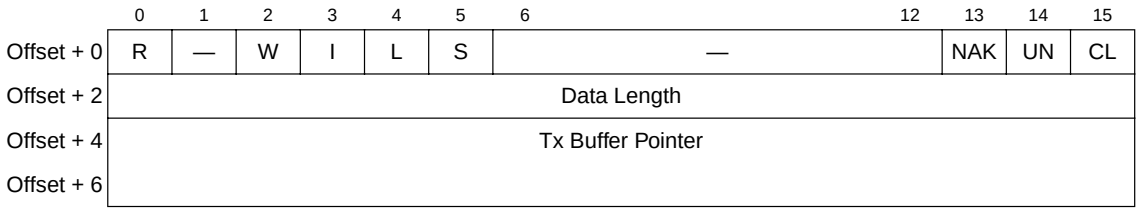


Figure 43-14. I2C TxBD

Table 43-10 describes I2C TxBD status and control bits.

Table 43-10. I2C TxBD Status and Control Bits

Bits	Name	Description
0	R	Ready. 0 The buffer is not ready to be sent. This BD or its buffer can be modified. The CP clears R after the buffer is sent or an error occurs. 1 The buffer is ready for transmission or is being sent. The BD cannot be modified once R is set.
1	—	Reserved and should be cleared.
2	W	Wrap (last BD in TxBD table). 0 Not the last BD in the table. 1 Last BD in the table. After this buffer is used, the CP transmits data using the BD pointed to by TBASE (top of the table). The number of BDs in this table is determined only by the W bit and overall space constraints of the dual-port RAM.
3	I	Interrupt. 0 No interrupt is generated after this buffer is serviced. 1 I2CER[TXB] or I2CER[TXE] is set when the buffer is serviced. If enabled, an interrupt occurs.
4	L	Last. 0 This buffer does not contain the last character of the message. 1 This buffer contains the last character of the message. The I2C controller generates a stop condition after sending this buffer.
5	S	Generate start condition. Provides ability to send back-to-back frames with one I2COM[STR] trigger. 0 Do not send a start condition before the first byte of the buffer. 1 Send a start condition before the first byte of the buffer. (Used to separate frames.) Note: If this BD is the first one in the frame when I2COM[STR] is triggered, a start condition is sent regardless of the value of TxBD[S].
6–12	—	Reserved and should be cleared.
13	NAK	No acknowledge. Indicates that the transmission was aborted because the last byte sent was not acknowledged. The I2C controller updates NAK after the buffer is sent.
14	UN	Underrun. Indicates that the I2C controller encountered a transmitter underrun condition while sending the associated buffer. The I2C controller updates UN after the buffer is sent.
15	CL	Collision. Indicates that transmission terminated because the transmitter was lost while arbitrating for the bus. The I2C controller updates CL after the buffer is sent.

Chapter 44
Parallel I/O Ports

The CPM supports four general-purpose I/O ports—ports A, B, C, and D. Each pin in the I/O ports can be configured as a general-purpose I/O signal or as a dedicated peripheral interface signal. Port C is unique in that 16 of its pins can generate interrupts to the interrupt controller.

Each pin can be configured as an input or output and has a latch for data output, read or written at any time, and configured as general-purpose I/O or a dedicated peripheral pin. Part of the pins can be configured as open-drain (the pin can be configured in a wired-OR configuration on the board). The pin drives a zero voltage but three-states when driving a high voltage.

Note that port pins do not have internal pull-up resistors. Due to the CPM's significant flexibility, many dedicated peripheral functions are multiplexed onto the ports. The functions are grouped to maximize the pins' usefulness in the greatest number of MPC8560 applications. The reader may not obtain a full understanding of the pin assignment capability described in this chapter without understanding the CPM peripherals.

44.1 Features

The following is a list of the parallel I/O ports' important features:

- Port A is 32 bits
- Port B is 28 bits
- Port C is 32 bits
- Port D is 28 bits
- All ports are bidirectional
- All ports have alternate on-chip peripheral functions
- All ports are three-stated at system reset
- All pin values can be read while the pin is connected to an on-chip peripheral
- Open-drain capability on some pins
- Port C offers 16 interrupt input pins

44.2 Port Registers

Each port has four memory-mapped, read/write, 32-bit control registers.

44.2.1 Port Open-Drain Registers (PODRA–PODRD)

The port open-drain register (PODR), shown in Figure 44-1, indicates a normal or wired-OR configuration of the port pins.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	OD0[1]	OD1[1]	OD2[1]	OD3[1]	OD4	OD5	OD6	OD7	OD8	OD9	OD10	OD11	OD12	OD13	OD14	OD15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_0D0C (PODRA), 0x9_0D2C (PODRB), 0x9_0D4C (PODRC), 0x9_0D6C (PODRD)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	OD16	OD17	OD18	OD19	OD20	OD21	OD22	OD23	OD24	OD25	OD26	OD27	OD28	OD29	OD30	OD31
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_0D0E (PODRA), 0x9_0D2E (PODRB), 0x9_0D4E (PODRC), 0x9_0D6E (PODRD)															

[1] These bits are valid for PODRA and PODRC only

Figure 44-1. Port Open-Drain Registers (PODRA–PODRD)

Table 44-1 describes PODR fields.

Table 44-1. PODR*x* Field Descriptions

Bits	Name	Description
0–31	OD*x*	Open-drain configuration. Determines whether the corresponding pin is actively driven as an output or is an open-drain driver. Note that bits OD0–OD3 are valid for PODRA and PODRC only. 0 The I/O pin is actively driven as an output. 1 The I/O pin is an open-drain driver. As an output, the pin is driven active-low, otherwise it is three-stated.

44.2.2 Port Data Registers (PDATA–PDATD)

A read of a port data register (PDAT*x*), shown in Figure 44-2, returns the data at the pin, independent of whether the pin is defined as an input or output. This allows detection of output conflicts at the pin by comparing the written data with the data on the pin.

A write to the PDAT*x* is latched and if the equivalent PDIR bit is configured as an output, the value latched for that bit is driven onto its respective pin. PDAT*x* can be read or written at any time and is not initialized.

If a port pin is selected as a general-purpose I/O pin, it can be accessed through the port data register (PDAT*x*). Data written to the PDAT*x* is stored in an output latch. If a port pin

is configured as an output, the output latch data is gated onto the port pin. In this case, when PDAT*x* is read, the port pin itself is read. If a port pin is configured as an input, data written to PDAT*x* is still stored in the output latch, but is prevented from reaching the port pin. In this case, when PDAT*x* is read, the state of the port pin is read.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	D0[1]	D1[1]	D2[1]	D3[1]	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
Reset	—															
R/W	R/W															
Addr	0x9_0D10 (PDATA), 0x9_0D30 (PDATB), 0x9_0D50 (PDATC), 0x9_0D70 (PDATD)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30	D31
Reset	—															
R/W	R/W															
Addr	0x9_0D12 (PDATA), 0x9_0D32 (PDATB), 0x9_0D52 (PDATC), 0x9_0D72 (PDATD)															

[1] These bits are valid for PDATA and PDATC only

Figure 44-2. Port Data Registers (PDATA–PDATD)

44.2.3 Port Data Direction Registers (PDIRA–PDIRD)

The port data direction register(PDIR), shown in Figure 44-3, is cleared at system reset.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	DR0[1]	DR1[1]	DR2[1]	DR3[1]	DR4	DR5	DR6	DR7	DR8	DR9	DR10	DR11	DR12	DR13	DR14	DR15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_0D00 (PDIRA), 0x9_0D20 (PDIRB), 0x9_0D40 (PDIRC), 0x9_0D60 (PDIRD)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DR16	DR17	DR18	DR19	DR20	DR21	DR22	DR23	DR24	DR25	DR26	DR27	DR28	DR29	DR30	DR31
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_0D02 (PDIRA), 0x9_0D22 (PDIRB), 0x9_0D42 (PDIRC), 0x9_0D62 (PDIRD)															

[1] These bits are valid for PDIRA and PDIRC only

Figure 44-3. Port Data Direction Register (PDIR)

Table 44-2 describes PDIR fields.

Table 44-2. PDIR Field Descriptions

Bits	Name	Description
0–31	DR*x*	Direction. Indicates whether a pin is used as an input or an output. Note that bits DR0–DR3 are valid for PDIRA and PDIRC only. 0 The corresponding pin is an input or is bidirectional. 1 The corresponding pin is an output.

44.2.4 Port Pin Assignment Register (PPAR)

The port pin assignment register (PPAR) is cleared at system reset.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	DD0[1]	DD1[1]	DD2[1]	DD3[1]	DD4	DD5	DD6	DD7	DD8	DD9	DD10	DD11	DD12	DD13	DD14	DD15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_0D04 (PPARA), 0x9_0D24 (PPARB), 0x9_0D44 (PPARC), 0x9_0D64 (PPARD)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	DD16	DD17	DD18	DD19	DD20	DD21	DD22	DD23	DD24	DD25	DD26	DD27	DD28	DD29	DD30	DD31
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_0D06 (PPARA), 0x9_0D26 (PPARB), 0x9_0D46 (PPARC), 0x9_0D66 (PPARD)															

[1] These bits are valid for PPARA and PPARC only

Figure 44-4. Port Pin Assignment Register (PPARA–PPARD)

Table 44-3 describes PPAR*x* fields.

Table 44-3. PPAR Field Descriptions

Bits	Name	Description
0–31	DD*x*	Dedicated enable. Indicates whether a pin is a general-purpose I/O or a dedicated peripheral pin. Note that bits DD0–DD3 are valid for PPARA and PPARC only. 0 General-purpose I/O. The peripheral functions of the pin are not used. 1 Dedicated peripheral function. The pin is used by the internal module. The on-chip peripheral function to which it is dedicated can be determined by other bits such as those is the PDIR.

44.2.5 Port Special Options Registers A–D (PSORA–PSORD)

Figure 44-5 shows the port special options registers (PSOR*x*).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	SO0[1]	SO1[1]	SO2[1]	SO3[1]	SO4	SO5	SO6	SO7	SO8	SO9	SO10	SO11	SO12	SO13	SO14	SO15
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_0D08 (PSORA), 0x9_0D28 (PSORB), 0x9_0D48 (PSORC), 0x9_0D68 (PSORD)															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Field	SO16	SO17	SO18	SO19	SO20	SO21	SO22	SO23	SO24	SO25	SO26	SO27	SO28	SO29	SO30	SO31
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_0D0A (PSORA), 0x9_0D2A (PSORB), 0x9_0D4A (PSORC), 0x9_0D6A (PSORD)															

[1] These bits are valid for PSORA and PSORC only

Figure 44-5. Special Options Registers (PSORA–POSRD)

PSOR bits are effective only if the corresponding PPAR*x*[DD*x*] = 1 (a dedicated peripheral function). Table 44-4 describes PSOR*x* fields.

Table 44-4. PSOR*x* Field Descriptions

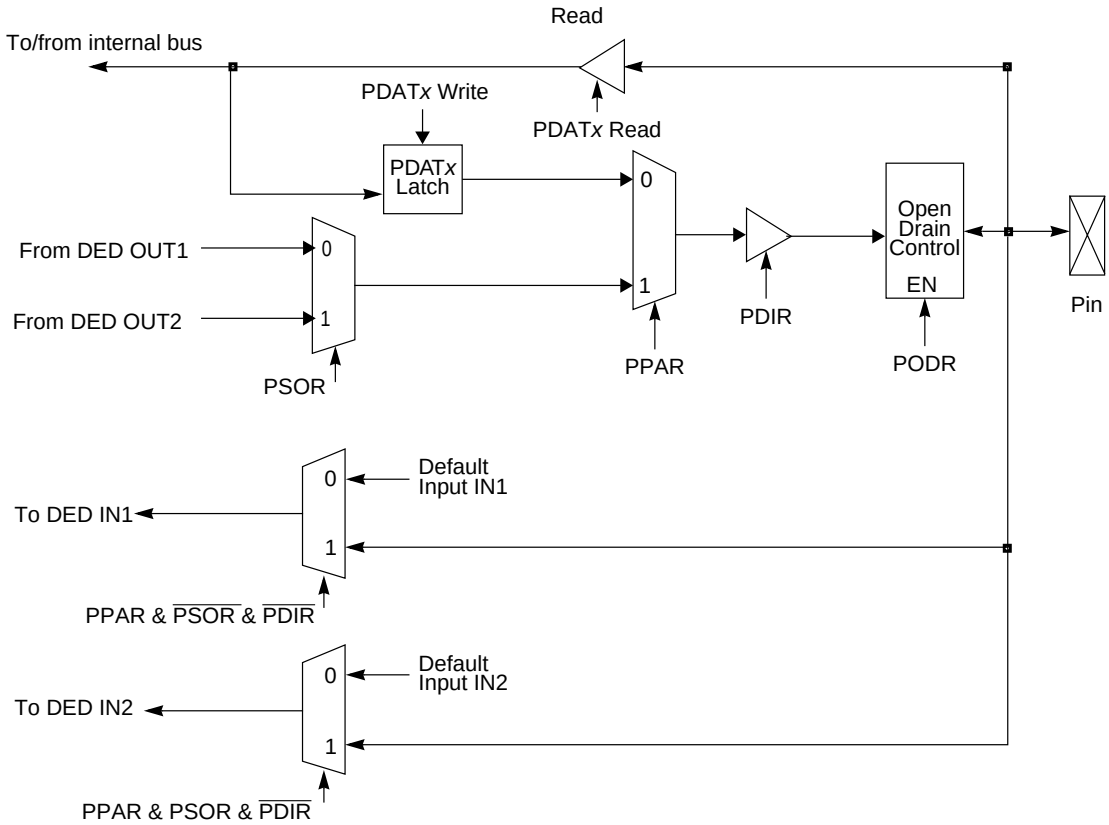
Bits	Name	Description
0–31	SO*x*	Special-option. Determines whether a pin configured for a dedicated function (PPAR*x*[DD*x*] = 1) uses option 1 or option 2. Note that bits SO0–SO3 are valid for PSORA and PSORC only. Options are described in Section 44.2, “Port Registers.” 0 Dedicated peripheral function. Option 1. 1 Dedicated peripheral function. Option 2.

NOTE

If the corresponding PPAR*x*[DD*x*] = 1 (configured as a general-purpose pin) before programming a PSOR*x* or PDIR*x* bit, a pin might function for a short period as an unwanted dedicated function and cause unknown behavior.

44.3 Port Block Diagram

Figure 44-6 shows the functional block diagram.



Register Name	0	1	Description
PPAR*x*	General purpose	Dedicated	Port pin assignment
PSOR*x*	Dedicated 1	Dedicated 2	Special operation
PDIR*x*	Input	Output	Direction [1]
PODR*x*	Regular	Open drain	
PDAT*x*	0	1	Data

[1] Bidirectional signals must be programmed as inputs (PDIR = 0).

Figure 44-6. Port Functional Operation

44.4 Port Pins Functions

Each pin can operate as a general purpose I/O pin or as a dedicated input or output pin.

44.4.1 General Purpose I/O Pins

Each one of the port pins is independently configured as a general-purpose I/O pin if the corresponding port pin assignment register (PPAR) bit is cleared. Each pin is configured as

a dedicated on-chip peripheral pin if the corresponding PPAR bit is set.When the port pin is configured as a general-purpose I/O pin, the signal direction for that pin is determined by the corresponding control bit in the port data direction register (PDIR). The port I/O pin is configured as an input if the corresponding PDIR bit is cleared; it is configured as an output if the corresponding PDIR bit is set. All PPAR and PDIR bits are cleared on total system reset, configuring all port pins as general-purpose input pins.

If a port pin is selected as a general-purpose I/O pin, it can be accessed through the port data register (PDAT*x*). Data written to the PDAT*x* is stored in an output latch. If a port pin is configured as an output, the output latch data is gated onto the port pin. In this case, when PDAT*x* is read, the port pin itself is read. If a port pin is configured as an input, data written to PDAT*x* is still stored in the output latch, but is prevented from reaching the port pin. In this case, when PDAT*x* is read, the state of the port pin is read.

44.4.2 Dedicated Pins

When a port pin is not configured as a general-purpose I/O pin, it has a dedicated functionality, as described in the following tables. Note that if an input to a peripheral is not supplied from a pin, a default value is supplied to the on-chip peripheral as listed in the right-most column.

NOTE

Some output functions can be output on 2 different pins. For example, the output for BRG1 can come out on both PC31 and PD19. The user can freely configure such functions to be output on two pins at once. However, there is typically no advantage in doing so unless there is a large fanout where it is advantageous to share the load between two pins.

Many input functions can also come from two different pins; see Section 44.5, “Ports Tables.”

44.5 Ports Tables

Table 44-5 through Table 44-8 describe the ports functionality according to the configuration of the port registers (PPAR*x*, PSOR*x*, and PDIR*x*). Each pin can function as a general purpose I/O, one of two dedicated outputs, or one of two dedicated inputs.

As shown in Figure 44-7, some input functions can come from two different pins for flexibility. Secondary option programming is relevant only if primary option is programmed to the default value.

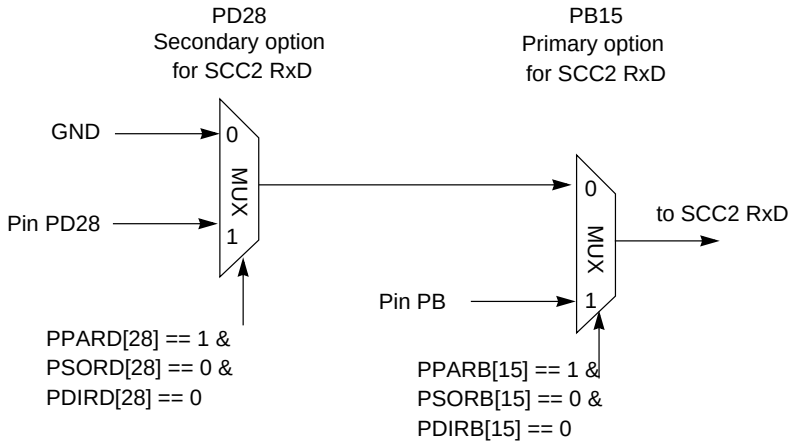


Figure 44-7. Primary and Secondary Option Programming

In the tables below, the default value for a primary option is simply a reference to the secondary option. In the secondary option, the programming is relevant only if the primary option is not used for the function.

Table 44-5 shows the port A pin assignments.

Table 44-5. Port A—Dedicated Pin Assignment (PPARA = 1)

Pin	Pin Function					
	PSORA = 0			PSORA = 1		
	PDIRA = 1 (Output)	PDIRA = 0 (Input)	Default Input	PDIRA = 1 (Output)	PDIRA = 0 (Input, or Inout if Specified)	Default Input
PA31	**FCC1: TxEnb** UTOPIA master	**FCC1: TxEnb** UTOPIA slave	GND		**FCC1: COL** MII	GND
PA30	**FCC1: TxClav** UTOPIA slave	**FCC1: TxClav** UTOPIA master **FCC1: TxClav0** MPHY, master, direct polling	GND	**FCC1: $\overline{\text{RTS}}$**	**FCC1: CRS** MII	GND
PA29	**FCC1: TxSOC** UTOPIA			**FCC1: TX_ER** MII		
PA28	**FCC1: RxEnb** UTOPIA master	**FCC1: RxEnb** UTOPIA slave	GND	**FCC1: TX_EN** MII/RMII		
PA27		**FCC1: RxSOC** UTOPIA	GND		**FCC1: RX_DV** MII/RMII	GND
PA26	**FCC1: RxClav** UTOPIA slave	**FCC1: RxClav** UTOPIA master **FCC1: RxClav0** MPHY, master, direct polling	GND		**FCC1: RX_ER** MII/RMII	GND

Table 44-5. Port A—Dedicated Pin Assignment (PPARA = 1) (continued)

Pin	Pin Function					
	PSORA = 0			PSORA = 1		
	PDIRA = 1 (Output)	PDIRA = 0 (Input)	Default Input	PDIRA = 1 (Output)	PDIRA = 0 (Input, or Inout if Specified)	Default Input
PA25	**FCC1: TxD[0]** UTOPIA 8 **FCC1: TxD[8]** **UTOPIA 16**			MSNUM[0] [1]		
PA24	**FCC1: TxD[1]** UTOPIA 8 **FCC1: TxD[9]** UTOPIA 16			MSNUM[1][1]		
PA23	**FCC1: TxD[2]** UTOPIA 8 **FCC1: TxD[10]** UTOPIA 16					
PA22	**FCC1: TxD[3]** UTOPIA 8 **FCC1: TxD[11]** UTOPIA 16					
PA21	**FCC1: TxD[4]** UTOPIA 8 **FCC1: TxD[12]** UTOPIA 16 **FCC1: TxD[3]** MII/HDLC nibble					
PA20	**FCC1: TxD[5]** UTOPIA 8 **FCC1: TxD[13]** UTOPIA 16 **FCC1: TxD[2]** MII/HDLC nibble					
PA19	**FCC1: TxD[6]** UTOPIA 8 **FCC1: TxD[14]** UTOPIA 16 **FCC1: TxD[1]** MII/HDLC nibble **FCC1: TxD[0]** RMII dibit					

Table 44-5. Port A—Dedicated Pin Assignment (PPARA = 1) (continued)

Pin	Pin Function					
	PSORA = 0			PSORA = 1		
	PDIRA = 1 (Output)	PDIRA = 0 (Input)	Default Input	PDIRA = 1 (Output)	PDIRA = 0 (Input, or Inout if Specified)	Default Input
PA18	**FCC1: TxD[7]** UTOPIA 8 **FCC1: TxD[15]** **UTOPIA 16** **FCC1: TxD[0]** MII/HDLC nibble **FCC1: TxD[0]** RMII dibit **FCC1: TxD** HDLC/transp					
PA17		**FCC1: RxD[7]** UTOPIA 8 **FCC1: RxD[15]** UTOPIA 16 **FCC1: RxD[0]** MII/HDLC nibble **FCC1: RxD[0]** RMII dibit **FCC1: RxD** HDLC/transp.	GND			
PA16		**FCC1: RxD[6]** UTOPIA 8 **FCC1: RxD[14]** UTOPIA 16 **FCC1: RxD[1]** MII/HDLC nibble **FCC1: RxD[1]** RMII dibit	GND			
PA15		**FCC1: RxD[5]** UTOPIA 8 **FCC1: RxD[13]** UTOPIA 16 **FCC1: RxD[2]** MII/HDLC nibble	GND			
PA14		**FCC1: RxD[4]** UTOPIA 8 **FCC1: RxD[12]** UTOPIA 16 **FCC1: RxD[3]** MII/HDLC nibble	GND			
PA13		**FCC1: RxD[3]** UTOPIA 8 **FCC1: RxD[11]** UTOPIA 16	GND	MSNUM[2][1]		

Table 44-5. Port A—Dedicated Pin Assignment (PPARA = 1) (continued)

Pin	Pin Function					
	PSORA = 0			PSORA = 1		
	PDIRA = 1 (Output)	PDIRA = 0 (Input)	Default Input	PDIRA = 1 (Output)	PDIRA = 0 (Input, or Inout if Specified)	Default Input
PA12		**FCC1: RxD[2]** UTOPIA 8 **FCC1: RxD[10]** UTOPIA 16	GND	MSNUM[3][1]		
PA11		**FCC1: RxD[1]** UTOPIA 8 **FCC1: RxD[9] FCC1** UTOPIA 16	GND	MSNUM[4][1]		
PA10		**FCC1: RxD[0]** UTOPIA 8 **FCC1: RxD[8]** UTOPIA 16	GND	MSNUM[5][1]		
PA9				**TDM_A1: L1TXD[0]** Output		GND
PA8	**FCC2: TxAddr[4]**				**TDM_A1: L1RXD[0]** Input, nibble **TDM_A1: L1RXD** Inout, serial	GND
PA7	**FCC2: TxAddr[3]**				**TDM_A1: L1TSYNC/GRANT**	GND
PA6	**FCC2: RxAddr[3]**				**TDM_A1: L1RSYNC**	GND
PA5	**SCC2: $\overline{\text{RSTRT}}$**	**FCC1: RxPrty** [2] UTOPIA (secondary option)	GND	**FCC2: RxAddr[2]** MPHY master		GND
PA4	**FCC2: RxAddr[1]** MPHY master	**SCC2: $\overline{\text{REJECT}}$**	VDD			VDD
PA3	**FCC2: RxAddr[0]** MPHY master	CLK19	GND		**TDM_A2: L1RXD[1]** Nibble	GND
PA2	**FCC2: TxAddr[0]** MPHY master	CLK20	GND			
PA1	**FCC2: TxAddr[1]** MPHY master	**SCC1: $\overline{\text{REJECT}}$**	VDD			VDD
PA0	**SCC1: $\overline{\text{RSTRT}}$**			**FCC2: TxAddr[2]** MPHY master		GND

[1] MSNUM[0–4] is the sub-block code of the peripheral controller using SDMA; MSNUM[5] indicates which section, transmit or receive, is active during the transfer. See Section 26.1.4, “Status Error Registers (SMSER and LMSER).”

[2] Available only when the primary option for this function is not used.

Table 44-6. shows the port B pin assignments.

Table 44-6. Port B Dedicated Pin Assignment (PPARB = 1)

Pin	Pin Function					
	PSORB = 0			PSORB = 1		
	PDIRB = 1 (Output)	PDIRB = 0 (Input)	Default Input	PDIRB = 1 (Output)	PDIRB = 0 (Input or Inout if Specified)	Default Input
PB31	**FCC2: TX_ER** MII	**FCC2: RxSOC** UTOPIA	GND		**TDM_B2: L1TXD** Inout	GND
PB30	**FCC2: TxSOC** UTOPIA	**FCC2: RX_DV** MII **FCC2: CRS_DV** RMII	GND		**TDM_B2: L1RXD** Inout	GND
PB29	**FCC2: RxClav** UTOPIA slave	**FCC2: RxClav** UTOPIA master	GND	**FCC2: TX_EN** MII/RMII	**TDM_B2: L1RSYNC**	GND
PB28	**FCC2: $\overline{\text{RTS}}$**	**FCC2: RX_ER** MII/RMII	GND	**SCC1: TXD**	**TDM_B2: L1TSYNC/GRANT**	GND
PB27	**FCC2: TxD[0]** UTOPIA 8	**FCC2: COL** MII	GND		**TDM_C2: L1TXD** Inout	GND
PB26	**FCC2: TxD[1]** UTOPIA 8	**FCC2: CRS** MII	GND		**TDM_C2: L1RXD** Inout	GND
PB25	**FCC2: TxD[4]** UTOPIA 8 **FCC2: TxD[3]** MII/HDLC nibble			**TDM_A1: L1TXD[3]** Nibble	**TDM_C2: L1TSYNC/GRANT**	GND
PB24	**FCC2: TxD[5]** UTOPIA 8 **FCC2: TxD[2]** MII/HDLC nibble	**TDM_A1: L1RXD[3]** Nibble	GND		**TDM_C2: L1RSYNC**	GND
PB23	**FCC2: TxD[6]** UTOPIA **FCC2: TxD[1]** MII/HDLC nibble **FCC2: TxD[1]** RMII dibit	**TDM_A1: L1RXD[2]** Nibble	GND		**TDM_D2: L1TXD** Inout	GND
PB22	**FCC2: TxD[7]** UTOPIA **FCC2: TxD[0]** MII/HDLC nibble **FCC2: TxD[0]** RMII dibit **FCC2: TxD** HDLC/transp. serial	**TDM_A1: L1RXD[1]** Nibble	GND		**TDM_D2: L1RXD** Inout	GND

Table 44-6. Port B Dedicated Pin Assignment (PPARB = 1) (continued)

Pin	Pin Function					
	PSORB = 0			PSORB = 1		
	PDIRB = 1 (Output)	PDIRB = 0 (Input)	Default Input	PDIRB = 1 (Output)	PDIRB = 0 (Input or Inout if Specified)	Default Input
PB21		**FCC2: RxD[7]** UTOPIA 8 **FCC2: RxD[0]** MII/HDLC nibble **FCC2: RxD[0]** RMII dibit **FCC2: RxD** HDLC/transp.. serial	GND	**TDM_A1: L1TXD[2]** Nibble	**TDM_D2: L1TSYNC/GRANT**	GND
PB20		**FCC2: RxD[6]** UTOPIA 8 **FCC2: RxD[1]** MII/HDLC nibble **FCC2: RxD[1]** RMII dibit	GND	**TDM_A1-L1TXD[1]** Nibble	**TDM_D2: L1RSYNC**	GND
PB19		**FCC2: RxD[5]** UTOPIA 8 **FCC2: RxD[2]** MII/HDLC nibble	GND	**TDM_D2: $\overline{\text{L1RQ}}$**	**TDM_A2: L1RXD[3]** Nibble	GND
PB18		**FCC2: RxD[4]** UTOPIA 8 **FCC2: RxD[3]** MII/HDLC nibble	GND	**TDM_D2: L1CLKO**	**TDM A2: L1RXD[2]** Nibble	GND
PB17	**TDM_A1: $\overline{\text{L1RQ}}$**	**FCC3: RX_DV** MII **FCC3: CRS_DV** RMII	GND		CLK17	GND
PB16	**TDM_A1: L1CLKO**	**FCC3: RX_ER** MII/RMII	GND		CLK18	GND
PB15	**FCC3: TX_ER** MII	**SCC2: RXD** (primary option)	by PD28		**TDM_C1: L1TXD** Inout (primary option)	by PD28
PB14	**FCC3: TX_EN** MII/RMII	**SCC3: RXD** (primary option)	by PD25		**TDM_C1: L1RXD** Inout (primary option)	by PD27
PB13	**TDM_B1: $\overline{\text{L1RQ}}$**	**FCC3: COL** MII	GND	**TDM_A2: L1TXD[1]** Nibble	**TDM_C1: L1TSYNC/GRANT** (primary option)	by PD16
PB12	**TDM_B1: L1CLKO**	**FCC3: CRS** MII	GND	**SCC2: TXD**	**TDM_C1: L1RSYNC** (primary option)	by PD26
PB11	**FCC2: TxD[0]** UTOPIA 8	**FCC3: RxD[3]** MII/HDLC nibble	GND		**TDM_D1: L1TXD** Inout (primary option)	by PD25

Table 44-6. Port B Dedicated Pin Assignment (PPARB = 1) (continued)

Pin	Pin Function					
	PSORB = 0			PSORB = 1		
	PDIRB = 1 (Output)	PDIRB = 0 (Input)	Default Input	PDIRB = 1 (Output)	PDIRB = 0 (Input or Inout if Specified)	Default Input
PB10	**FCC2: TxD[1]** UTOPIA 8	**FCC3: RxD[2]** MII/HDLC nibble	GND		**TDM_D1: L1RXD** Inout (primary option)	by PD24
PB9	**FCC2: TxD[2]** UTOPIA 8	**FCC3: RxD[1]** MII/HDLC nibble **FCC3: RxD[1]** RMII dibit	GND	**TDM_A2: L1TXD[2]** Nibble	**TDM_D1: L1TSYNC/GRANT** (primary option)	by PD4
PB8	**FCC2: TxD[3]** UTOPIA 8	**FCC3: RxD[0]** MII/HDLC nibble **FCC3: RxD[0]** RMII dibit **FCC3: RxD** HDLC/transp. serial	GND	**SCC3: TXD**	**TDM_D1: L1RSYNC** (primary option)	by PD23
PB7	**FCC3: TXD[0]** MII/HDLC nibble **FCC3: TXD[0]** RMII dibit **FCC3: TXD** HDLC/transp. serial	**FCC2: RxD[3]** UTOPIA 8 (primary option)	by PC10	**TDM_A2: L1TXD[0]** Output, nibble	**TDM_A2: L1TXD** Inout, serial (primary option)	by PD22
PB6	**FCC3: TXD[1]** MII/HDLC nibble **FCC3: TXD[1]** RMII dibit	**FCC2: RxD[2]** UTOPIA 8 (primary option)	by PC11		**TDM_A2: L1RXD** Inout, serial **TDM_A2: L1RXD[0]** Input, nibble (primary option)	by PD21
PB5	**FCC3: TXD[2]** MII/HDLC nibble	**FCC2: RxD[1]** UTOPIA 8 (primary option)	by PD10		**TDM_A2: L1TSYNC/GRANT** (primary option)	by PC9
PB4	**FCC3: TXD[3]** MII/HDLC nibble	**FCC2: RxD[0]** UTOPIA 8 (primary option)	by PD11	**FCC3: RTSN**	**TDM_A2: L1RSYNC** (primary option)	by PD20

Table 44-7 shows the port C pin assignments.

Table 44-7. Port C Dedicated Pin Assignment (PPARC = 1)

PIN	Pin Function					
	PSORC = 0			PSORC = 1		
	PDIRC = 1 (Output)	PDIRC = 0 (Input)	Default Input	PDIRC = 1 (Output)	PDIRC = 0 (Input or Inout if Specified)	Default Input
PC31	**BRG1: BRGO**	CLK1	CLK5			
PC30	**FCC2: TxD[3]** UTOPIA 8	CLK2	CLK6	**Timer1:$\overline{\text{TOUT}}$**		
PC29	**BRG2: BRGO**	CLK3/TIN2	CLK7		**SCC1: $\overline{\text{CTS}}$** [1] (secondary option)	GND
PC28	**Timer2: $\overline{\text{TOUT}}$**	CLK4/TIN1	CLK8	**FCC2: RxAddr[4]**	**SCC2: $\overline{\text{CTS}}$**[1] (secondary option)	GND
PC27	**FCC3: TxD** HDLC/transp. serial **FCC3: TxD[0]** MII/HDLC nibble **FCC3: TxD[0]** RMII dibit	CLK5	GND	**BRG3: BRGO**		
PC26	**Timer3: $\overline{\text{TOUT}}$**	CLK6	GND		TMCLK real-time counter	BRGO1
PC25	**FCC2: TxD[2]** UTOPIA 8	CLK7	GND	**BRG4: BRGO**		
PC24	**FCC2: TxD[3]** UTOPIA 8	CLK8	GND	**Timer4: $\overline{\text{TOUT}}$**		
PC23	**BRG5: BRGO**	CLK9	CLK13			
PC22	**FCC1: TxPrty** UTOPIA (secondary option) [2]	CLK10	CLK14			
PC21	**BRG6: BRGO**	CLK11	CLK15			
PC20		CLK12	CLK16		**timer1/2: $\overline{\text{TGATE1}}$**	GND
PC19	**BRG7: BRGO**	CLK13	GND		**SPI: SPICLK**[2] Inout (secondary option)	
PC18		CLK14	GND		**timer3/4: $\overline{\text{TGATE2}}$**	GND
PC17	**BRG8: BRGO**	CLK15/TIN3	GND			
PC16		CLK16/TIN4	GND			
PC15		**SCC1: $\overline{\text{CTS}}$** (primary option)	by PC5	**FCC1: TxAddr[0]** MPHY, master	**FCC1: TxAddr[0]** [3] MPHY, slave **FCC2: TxAddr[4]** MPHY, slave	GND

Table 44-7. Port C Dedicated Pin Assignment (PPARC = 1) (continued)

PIN	Pin Function					
	PSORC = 0			PSORC = 1		
	PDIRC = 1 (Output)	PDIRC = 0 (Input)	Default Input	PDIRC = 1 (Output)	PDIRC = 0 (Input or Inout if Specified)	Default Input
PC14		**SCC1: $\overline{\text{CD}}$**	GND	**FCC1: RxAddr[0]** MPHY, master	**FCC1: RxAddr[0]**[3] MPHY, slave **FCC2: RxAddr[4]** MPHY, slave	GND
PC13	**TDM_D1: $\overline{\text{L1RQ}}$**	**SCC2: $\overline{\text{CTS}}$** (primary option)	by PC4	**FCC1: TxAddr[1]** MPHY, master	**FCC1: TxAddr[1]**[3] MPHY, slave **FCC2: TxAddr[3]** MPHY, slave	GND
PC12	**SI1: L1ST3**	**SCC2: $\overline{\text{CD}}$**	GND	**FCC1: RxAddr[1]** MPHY, master	**FCC1: RxAddr[1]**[3] MPHY, slave **FCC2: RxAddr[3]** MPHY, slave	GND
PC11	**TDM_D1: L1CLKO**	**SCC3: $\overline{\text{CTS}}$** (primary option)	by PC8	**TDM_A2: L1TXD[3]** Nibble	**FCC2: RxD[2]**[1] UTOPIA 8 (secondary option)	GND
PC10	**FCC1: TxD[2]** UTOPIA 16	**SCC3: $\overline{\text{CD}}$**	GND	**SI1: L1ST4** strobe	**FCC2: RxD[3]**[1] UTOPIA (secondary option)	GND
PC9	**FCC1: TxD[1]** UTOPIA 16	**SCC4: $\overline{\text{CTS}}$** (primary option)	by PC3	**SI2: L1ST1** strobe	**TDM_A2: L1TSYNC/GRANT**[1] (secondary option)	GND
PC8	**FCC1: TxD[0]** UTOPIA 16	**SCC4: $\overline{\text{CD}}$**	GND	**SI2: L1ST2** Strobe	**SCC3: $\overline{\text{CTS}}$**[1] (secondary option)	GND
PC7	**TDM_C1: $\overline{\text{L1RQ}}$**	**FCC1: $\overline{\text{CTS}}$**	GND	**FCC1: TxAddr[2]** MPHY master, multiplexed: polling	**FCC1: TxAddr[2]**[3] MPHY, slave, multiplexed polling **FCC1: TxClav1**[3] MPHY, master, direct polling **FCC2: TxAddr[2]** MPHY, slave, multiplexed polling	GND
PC6	**TDM_C1: L1CLKO**	**FCC1: $\overline{\text{CD}}$**	GND	**FCC1: RxAddr[2]** MPHY, master, multiplexed polling	**FCC1: RxAddr[2]**[3] MPHY, slave, multiplexed polling) **FCC1: RxClav1**[3] MPHY, master, direct polling **FCC2: RxAddr[2]** MPHY, slave, multiplexed polling	GND
PC5	**FCC2: TxClav** UTOPIA, slave	**FCC2: TxClav** UTOPIA, master	GND	**SI2: L1ST3** Strobe	**FCC2: $\overline{\text{CTS}}$**	GND

Table 44-7. Port C Dedicated Pin Assignment (PPARC = 1) (continued)

PIN	Pin Function					
	PSORC = 0			PSORC = 1		
	PDIRC = 1 (Output)	PDIRC = 0 (Input)	Default Input	PDIRC = 1 (Output)	PDIRC = 0 (Input or Inout if Specified)	Default Input
PC4	**FCC2: RxEnb** UTOPIA, master	**FCC2: RxEnb** UTOPIA, slave	GND	**SI2: L1ST4** Strobe	**FCC2: $\overline{\text{CD}}$**	GND
PC3	**FCC2: TxD[2]** UTOPIA 8	**FCC3: $\overline{\text{CTS}}$**	GND		**SCC4: $\overline{\text{CTS}}$**[1] (secondary option)	GND
PC2	**FCC2: TxD[3]** UTOPIA 8	**FCC3: $\overline{\text{CD}}$**	GND			
PC1	**BRG6: BRGO**		GND	**TDM_A2: $\overline{\text{L1RQ}}$**	**SPI: SPICLK**[2] Inout (secondary option)	
PC0	**BRG7: BRGO**		GND	**TDM_A2: L1CLKO**		GND

1 Available only when the primary option for this function is not used.

2 Available only when the primary option for this function is not used.

3 MPHY Address pins 3,4 (master mode) can come from FCC2, depending on CMXUAR programming. (See Section 23.4.1, “CMX UTOPIA Address Register (CMXUAR).”)

Table 44-8 shows the port D pin assignments.

Table 44-8. Port D Dedicated Pin Assignment (PPARD = 1)

Pin	Pin Function					
	PSORD = 0			PSORD = 1		
	PDIRD = 1 (Output)	PDIRD = 0 (Input)	Default Input	PDIRD = 1 (Output)	PDIRD = 0 (Input, or Inout if Specified)	Default Input
PD31		**SCC1: RXD**	GND			
PD30	**FCC2: TxEnb** UTOPIA master	**FCC2: TxEnb** UTOPIA slave	GND	**SCC1: TXD**		
PD29	**SCC1: $\overline{\text{RTS}}$**			**FCC1: RxAddr[3]** [1] MPHY, master, multiplexed polling **FCC2: RxAddr[4]** MPHY, master, multiplexed polling	**FCC1: RxAddr[3]** [2] MPHY, slave, multiplexed polling **FCC1: RxClav2** [2] MPHY, master, direct polling **FCC2: RxAddr[1]** MPHY, slave, multiplexed polling	GND
PD28	**FCC1: TxD[7]** UTOPIA 16 bit	**SCC2: RXD** [3] (secondary option)	GND		**TDM_C1: L1TXD**[3] Inout (secondary option)	GND

Table 44-8. Port D Dedicated Pin Assignment (PPARD = 1) (continued)

Pin	Pin Function					
	PSORD = 0			PSORD = 1		
	PDIRD = 1 (Output)	PDIRD = 0 (Input)	Default Input	PDIRD = 1 (Output)	PDIRD = 0 (Input, or Inout if Specified)	Default Input
PD27	**SCC2: TXD**	**FCC1: RxD[7]** UTOPIA 16	GND		**TDM_C1: L1RXD**[3] Inout (secondary option)	GND
PD26	**SCC2: $\overline{\text{RTS}}$**	**FCC1: RxD[6]** UTOPIA 16	GND		**TDM_C1: L1RSYNC**[3] (secondary option)	GND
PD25	**FCC1: TxD[6]** UTOPIA 16	**SCC3: RXD**[3] (secondary option)	GND		**TDM_D1: L1TXD**[3] Inout (secondary option)	GND
PD24	**SCC3: TXD**	**FCC1: RxD[5]** UTOPIA 16	GND		**TDM_D1: L1RXD**[3] Inout (secondary option)	GND
PD23	**SCC3: $\overline{\text{RTS}}$**	**FCC1: RxD[4]** UTOPIA 16	GND		**TDM_D1: L1RSYNC**[3] (secondary option)	GND
PD22	**FCC1: TxD[5]** UTOPIA 16	**SCC4: RXD**	GND	**TDM_A2: L1TXD[0]**[3] Output, nibble (secondary option)	**TDM_A2: L1TXD**[3] Inout, serial (secondary option)	GND
PD21	**SCC4: TXD**	**FCC1: RxD[3]** UTOPIA 16	GND		**TDM_A2: L1RXD**[3] Inout, serial **TDM_A2: L1RXD[0]**[3] Input, nibble (secondary option)	GND
PD20	**SCC4: $\overline{\text{RTS}}$**	**FCC1: RxD[2]** UTOPIA 16	GND		**TDM_A2: L1RSYNC**[3] (secondary option)	GND
PD19	**FCC1: TxAddr[4]** [1] MPHY, master, multiplexed polling **FCC2: TxAddr[3]** MPHY, master, multiplexed polling	**FCC1: TxAddr[4]** [2] MPHY, slave, multiplexed polling **FCC1: TxClav3**[2] MPHY, master, direct polling **FCC2: TxAddr[0]** MPHY, slave, multiplexed polling	GND	**BRG1: BRGO**	**SPI: $\overline{\text{SPISEL}}$** (primary option)	V_{DD}
PD18	**FCC1: RxAddr[4]**[1] MPHY, master, multiplexed polling **FCC2: RxAddr[3]** MPHY, master, multiplexed polling	**FCC1: RxAddr[4]**[2] MPHY, slave, multiplexed polling **FCC1: RxClav3**[2] MPHY, master, direct polling **FCC2: RxAddr[0]** MPHY, slave, multiplexed polling	GND		**SPI: SPICLK** Inout (primary option)	GND

Table 44-8. Port D Dedicated Pin Assignment (PPARD = 1) (continued)

Pin	Pin Function					
	PSORD = 0			PSORD = 1		
	PDIRD = 1 (Output)	PDIRD = 0 (Input)	Default Input	PDIRD = 1 (Output)	PDIRD = 0 (Input, or Inout if Specified)	Default Input
PD17	**BRG2: BRGO**	**FCC1: RxPrty** UTOPIA (primary option)	GND		**SPI: SPIMOSI** Inout	V_{DD}
PD16	**FCC1: TxPrty** UTOPIA (primary option)	**TDM_C1: L1TSYNC/GRANT**[3] (secondary option)	GND		**SPI: SPIMISO** Inout	SPIMOSI
PD15	**TDM_C2: $\overline{\text{L1RQ}}$**	**FCC1: RxD[1]** UTOPIA 16	GND		**I2C: I2CSDA** Inout	V_{DD}
PD14	**TDM_C2: L1CLKO**	**FCC1: RxD[0]** UTOPIA 16	GND		**I2C: I2CSCL** Inout	GND
PD13	**SI1: L1ST1**				**TDM_B1: L1TXD** Inout	GND
PD12	**SI1: L1ST2**				**TDM_B1: L1RXD** Inout	GND
PD11	**TDMB2: $\overline{\text{L1RQ}}$**	**FCC2: RxD[0]**[3] UTOPIA 8 (secondary option)	GND		**TDM_B1: L1TSYNC/GRANT**	GND
PD10	**TDMB2: L1CLKO**	**FCC2: RxD[1]**[3] UTOPIA 8 (secondary option)	GND	**BRG4: BRGO**	**TDM_B1: L1RSYNC**	GND
PD9				**BRG3: BRGO**	**FCC2: RxPrty** UTOPIA	GND
PD8	**FCC2: TxPrty** UTOPIA		GND	**BRG5: BRGO**		
PD7				**FCC1: TxAddr[3]**[1] MPHY, master, multiplexed polling **FCC2: TxAddr[4]** MPHY, master, multiplexed polling	**FCC1: TxAddr[3]**[2] MPHY, slave, multiplexed polling **FCC1: TxClav2**[2] MPHY, master, direct polling **FCC2: TxAddr[1]** MPHY, slave, multiplexed polling	GND
PD6	**FCC1: TxD[4]** UTOPIA 16					
PD5	**FCC1: TxD[3]** UTOPIA 16					
PD4	**BRG8: BRGO**	**TDM_D1: L1TSYNC/GRANT**[3] (secondary option)	GND	**FCC3: $\overline{\text{RTS}}$**		GND

1 MPHY address pins 3 and 4 (master mode) can come from FCC2, depending on CMXUAR programming. (See Section 23.4.1, “CMX UTOPIA Address Register (CMXUAR).”)

2 MPHY address pins 0–4 (slave mode) can come from FCC2, depending on CMXUAR programming. (See Section 23.4.1, “CMX UTOPIA Address Register (CMXUAR).”)

3 Available only when the primary option for this function is not used.

44.6 Interrupts from Port C

The port C lines associated with $\overline{\text{CD}}x$ and $\overline{\text{CTS}}x$ have a mode of operation where the pin can be internally connected to the SCC/FCC but can also generate interrupts. Port C still detects changes on the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins and asserts the corresponding interrupt request, but the SCC/FCC simultaneously uses $\overline{\text{CTS}}$ and/or $\overline{\text{CD}}$ to automatically control operation. This lets the user fully implement protocols V.24, X.21, and X.21 bis (with the assistance of other general-purpose I/O lines).

To configure a port C pin as a $\overline{\text{CTS}}$ or $\overline{\text{CD}}$ pin that connects to the SCC/FCC and generates interrupts, these steps should be followed:

1. Write the corresponding PPARC bit with a 1 and PSORC bit with 0.
2. Write the corresponding PDIRC bit with a zero.
3. Set the SIEXR bit (in the interrupt controller) to determine which edges cause interrupts.
4. Write the corresponding SIMR (mask register) bit with a 1 to allow interrupts to be generated to the core.
5. The pin value can be read at any time using PDATC.

NOTE

After connecting $\overline{\text{CTS}}$ or $\overline{\text{CD}}$ to the SCC/FCC, the user must also choose the normal operation mode in GSMR[DIAG] to enable and disable SCC/FCC transmission and reception with these pins.

Chapter 45
CPM ROM Microcode: SS7 Controller

45.1 SS7 Data Structure Organization

The SS7 controller uses the same data structures as MCC. See Chapter 33, “Multi-Channel Controllers (MCCs),” in the *MPC8560 User’s Manual*.

45.2 Global SS7 Parameters

The global parameters of SS7 Controller are identical to MCC global parameters. See Chapter 33, “Multi-Channel Controllers (MCCs),” in the *MPC8560 User’s Manual*.

45.3 Channel Extra Parameters

The extra parameters of SS7 channels are identical to the MCC channel extra parameters. See Chapter 33, “Multi-Channel Controllers (MCCs),” in the *MPC8560 User’s Manual*.

45.4 Super Channel Table

The super channel table of SS7 Controller is identical to MCC super channel table. See Chapter 33, “Multi-Channel Controllers (MCCs),” in the *MPC8560 User’s Manual*.

45.5 TDM Mapping

The mapping of SS7 channels to TDM channels is identical to mapping of MCC channels. See Chapter 33, “Multi-Channel Controllers (MCCs),” in the *MPC8560 User’s Manual*.

45.6 SS7 Commands

The commands of the SS7 controller are identical to MCC commands. See Chapter 33, “Multi-Channel Controllers (MCCs),” in the *MPC8560 User’s Manual*.

45.7 SS7 Exceptions

The SS7 exceptions are almost identical to MCC exceptions. See Chapter 33, “Multi-Channel Controllers (MCCs),” in the *MPC8560 User’s Manual*. The differences are in additional indications in the interrupt table entries. The SS7 interrupt table entry is described in Section 45.7.1, “Interrupt Table Entry.

45.7.1 Interrupt Table Entry

Each interrupt table entry, shown in Figure 45-1, contains information about channel-specific events. Transmit interrupt table entries show only events caused by transmission; receive interrupt table entries shows only events caused by reception.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	V	W	OCT	SUERM	FISU		UN	TXB	—	AERM	NID	IDL	MRF	RXF	BSY	RXB
Reset																
R/W	R/W															

	16	17	18	19	20	21	22	23	24	25	26	31
Field	—		Channel Number									
Reset												
R/W	R/W											

Note that items in bold should be initialized by the user.

Figure 45-1. Interrupt Table Entry

Table 45-1 describes interrupt table fields.

Table 45-1. Interrupt Table Entry Field Descriptions

Bits	Name	Description
0	V	Valid bit. V = 1 indicates that this entry contains valid interrupt information. Upon generating a new entry, the CP sets V = 1. The user clears V immediately after it reads the interrupt flags of the entry (before processing the interrupt). The V bits in the table are user-initialized. During initialization, the user must clear the valid bit for each table entry.
1	W	Wrap bit. W = 1 indicates the last interrupt table entry. The next event’s entry is written/read (by CP/user) from the address contained in INTBASE. During initialization, the user must clear all W bits in the table except for the last one which must be set.
2	OCT	N octets received. If the channel is in octet counting, this bit is set when N octets have been received.
3	SUERM	SU error monitor threshold reached. The SU error monitor has reached the programmed threshold T.
4	FISU	FISU transmission started. The CP has started automatic FISU transmission if the first BD of frame does not have its ready bit set and the SEN_FISU bit is enabled in SS7_OPT register. Please refer to SEN_FISU bit in 44.6.4, SS7 Configuration Register (SS7_OPT).
5	—	Reserved, should be cleared during initialization.
6	UN	Tx no data. The CP sets this flag if there is no data available to be sent to the transmitter. The transmitter sends an ABORT indication and then sends idles.

Table 45-1. Interrupt Table Entry Field Descriptions (continued)

Bits	Name	Description
7	TXB	Tx buffer. A buffer has been completely transmitted. TXB is set (and an interrupt request is generated) as soon as the programmed number of PAD characters (or the closing flag, for PAD = 0) is written to MCC transmit FIFO. This controls when the TXB interrupt is given in relation to the closing flag sent out at TXD. Section 45.8.2, “SS7 Transmit Buffer Descriptor (TxBD) describes how PAD characters are used.
8	—	Reserved, should be cleared during initialization.
9	AERM	The alignment error rate monitor has reached the programmed threshold M.
10	NID	Set whenever a pattern that is not an idle pattern is identified.
11	IDL	Idle. Set when the channel’s receiver identifies the first occurrence of SS7 idle (0xFFFE) after any non-idle pattern.
12	MRF	Maximum receive frame length violation. This interrupt occurs in SS7 mode when more bytes are received than the value specified in MFLR. This interrupt is generated as soon as the MFLR value is exceeded; the remainder of the frame is discarded
13	RXF	Rx frame. A complete SS7 frame (signal unit) has been received.
14	BSY	Busy. A frame was received but was discarded due to lack of buffers.
15	RXB	Rx buffer. A buffer has been received on this channel that was not the last buffer in frame.
16–18	—	Reserved, should be cleared during initialization.
19–26	CN	Channel number. Identifies the channel index (0–255) of the requesting channel.
27–31	—	Reserved, should be cleared during initialization.

45.8 SS7 Buffer Descriptors

The SS7 buffer descriptors (BDs) are almost identical to MCC buffer descriptors. See Chapter 33, “Multi-Channel Controllers (MCCs),” in the *MPC8560 User’s Manual*. The SS7 RxBDs have an additional status flag for short frames (RxBD[SF]), and the SS7 TxBDs have a retransmit control bit (TxBD[RT]). The following sections describe the BD structures for SS7 operation.

45.8.1 SS7 Receive Buffer Descriptor (RxBD)

Figure 45-2 shows the SS7 RxBD.

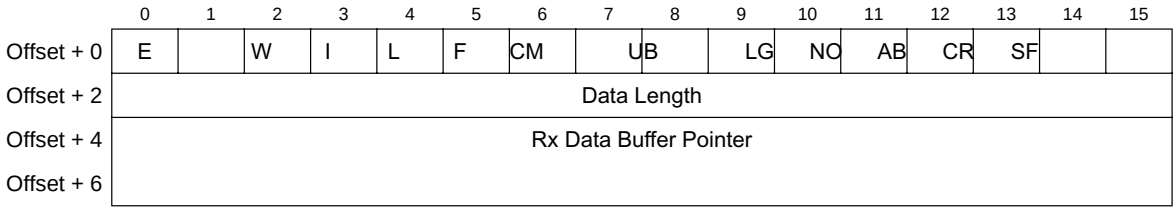


Figure 45-2. SS7 Receive Buffer Descriptor (RxBD)

SS7 RxBD fields are described in Table 45-2.

Table 45-2. SS7 RxBD Field Descriptions

Bits	Name	Description
0	E	Empty 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 user is free to examine or write to any fields of this RxBD. The CP does not use this BD again while the empty bit remains zero. 1 The data buffer associated with this BD is empty, or reception is in progress. This RxBD and its associated receive buffer are in use by the CP. When E = 1, the user should not write any fields of this RxBD.
1		Reserved, should be cleared during initialization.
2	W	Wrap (final BD in table) 0 This is not the last BD in the RxBD table. 1 This is the last BD in the RxBD table. After this buffer has been used, the CP receives incoming data into the first BD in the table (the BD pointed to by RBASE). The number of RxBDs in this table is programmable and is determined by the wrap bit.
3	I	Interrupt 0 The RXB bit is not set after this buffer has been used, but RXF operation remains unaffected. 1 The RXB or RXF bit in the HDLC interrupt circular table entry is set when this buffer has been used by the HDLC controller. These two bits may cause interrupts (if enabled).
4	L	Last in frame (only for HDLC mode of operation). The HDLC controller sets L = 1, when this buffer is the last in a frame. This implies the reception either of a closing flag or of an error, in which case one or more of the CD, OV, AB, and LG bits are set. The HDLC controller writes the number of frame octets to the data length field. 0 This buffer is not the last in a frame. 1 This buffer is the last in a frame.
5	F	First in frame. The HDLC controller sets F = 1 for the first buffer in a frame. In transparent mode, F indicates that there was a synchronization before receiving data in this BD. 0 This is not the first buffer in a frame. 1 This is the first buffer in a frame.
6	CM	Continuous mode 0 Normal operation (The empty bit (bit 0) is cleared by the CP after this BD is closed). 1 The empty bit (bit 0) is not cleared by the CP after this BD is closed, allowing the associated data buffer to be overwritten automatically when the CP next accesses this BD. However, if an error occurs during reception, the empty bit is cleared regardless of the CM bit setting.
7		Reserved, should be cleared during initialization.
8	UB	User bit. UB is a user-defined bit that the CPM never sets nor clears. The user determines how this bit is used.
9		Reserved, should be cleared during initialization.
10	LG	Rx frame length violation (HDLC mode only). Indicates that a frame length greater than the maximum value was received in this channel. Only the maximum-allowed number of bytes, MFLR rounded to the nearest higher word alignment, are written to the data buffer. This event is recognized as soon as the MFLR value is exceeded when data is word-aligned. When data is not word-aligned, this interrupt occurs when the SDMA writes 64 bits to memory. The worst-case latency from MFLR violation until detected is 7 bytes timing for this channel. When MFLR violation is detected, the receiver is still receiving even though the data is discarded. The buffer is closed upon detecting a flag, and this is considered to be the closing flag for this buffer. At this point, LG is set (1) and an interrupt may be generated. The length field for this buffer is everything between the opening flag and this last identifying flag.

Table 45-2. SS7 RxBD Field Descriptions (continued)

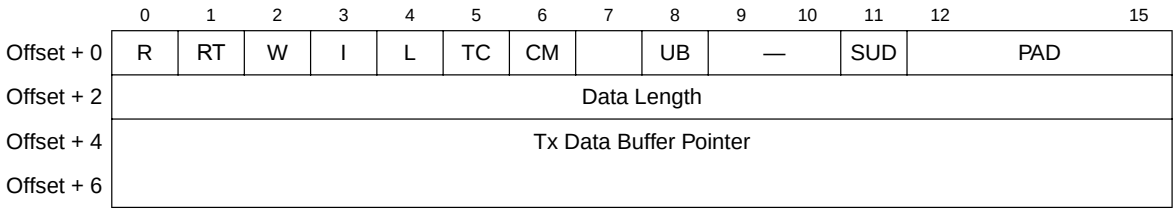
Bits	Name	Description
11	NO	Rx nonoctet-aligned frame. A frame of bits not divisible exactly by eight was received. NO = 1 for any type of nonalignment regardless of frame length. The shortest frame that can be detected is of type FLAG-BIT-FLAG, which causes the buffer to be closed with NO error indicated. The following shows how the nonoctet alignment is reported and where data can be found. msb lsb xxx..... xx | 1 | 000..... 0 Valid data | Invalid data To accommodate the extra word of data that may be written at the end of the frame, it is recommended to reserve MFLR + 8 bytes for each buffer data.
12	AB	Rx abort sequence. A minimum of seven consecutive 1s was received during frame reception. Abort is not detected between frames. The sequence Closing-Flag, data, CRC, AB, data, opening-flag... does not cause an abort error. If the abort is long enough to be an idle, an idle line interrupt may be generated. An abort within the frame is not reported by a unique interrupt but rather with a RXF interrupt and the user has to examine the BD.
13	CR	Rx CRC error. Set if this frame contains a CRC error. The received CRC bytes are always written to the receive buffer.
14	SF	Short frame indication. Set if the received frame is less than 5 octets.
15		Reserved, should be cleared during initialization.

The data length and buffer pointer are described as follows:

- Data length. Data length is the number of octets written by the CP into this BD’s data buffer. It is written by the CP when the BD is closed. When this is the last BD in the frame (L = 1), the data length contains the total number of frame octets (including two or four bytes for CRC). Note that memory allocated for buffers should be not smaller than the contents of the maximum receive buffer length register (MRBLR). The data length does not include the time stamp.
- Rx buffer pointer. The receive buffer pointer points to the first location of the associated data buffer. This value must be equal to 8*n if ECHAMR[TS]=0 and equal to 8*n-4 if ECHAMR[TS]=1 (n = any integer larger than 0).

45.8.2 SS7 Transmit Buffer Descriptor (TxBD)

Figure 45-3 shows the SS7 TxBD.



Items in bold must be initialized by the user.

Figure 45-3. SS7 Transmit Buffer Descriptor (TxBD)

Table 45-3 describes SS7 TxBD fields.

Table 45-3. SS7 TxBD Field Descriptions

Bits	Name	Description
0	R	Ready 0 The buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The CP clears this bit after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer is ready to be transmitted. The transmission may have begun, but it has not completed. The user cannot modify this BD once this bit is set.
1	RT	Retransmit 0 Normal operation 1 The CP will repeat transmission of this BD until the RT bit is cleared. After the RT bit is cleared the CP advances to the next BD in the table. This feature is useful for automatic LSSU retransmission.
2	W	Wrap (final BD in table) 0 This is not the last BD in the TxBD table. 1 This is the last BD in the TxBD table. After this buffer is used, the CP receives incoming data into the first BD in the table (the BD pointed to by TBASE). The number of TxBDs in this table is programmable and is determined the wrap bit.
3	I	Interrupt 0 No interrupt is generated after this buffer has been serviced. 1 TXB in the interrupt table entry is set when this buffer has been serviced by the MCC. This bit can cause an interrupt (if enabled).
4	L	Last 0 This is not the last buffer in the frame. 1 This is the last buffer in the current frame.
5	TC	Tx CRC. Valid only when L = 1. Otherwise it must be ignored. 0 Transmit the closing flag after the last data byte. This setting can be used for testing purposes to send an erroneous CRC after the data. 1 Transmit the CRC sequence after the last data byte.
6	CM	Continuous mode 0 Normal operation. 1 The CP does not clear the ready bit after this BD is closed, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this BD. However, the R bit is cleared if an error occurs during transmission, regardless of the CM bit setting.
7	—	Reserved, should be cleared during initialization.

Table 45-3. SS7 TxBD Field Descriptions (continued)

Bits	Name	Description
8	UB	User bit. UB is a user-defined bit that the CPM never sets nor clears. The user determines how this bit is used.
9–10	—	Reserved, should be cleared.
11	SUD	Signal unit delay 0 This buffer does not have a transmission delay. 1 A time delay of JTTDelay x 512 µs passes before this buffer is transmitted. Can be used for LSSU transmission according to the JT Q.703 Standard which defines a 24 ms delay between back-to-back LSSUs. This bit is only valid when SS7_OPT[STD] is set.
12–15	PAD	Pad characters. These four bits indicate the number of PAD characters (0x7E or 0xFF depending on the IDLM mode selected in the ECHAMR register) that the transmitter sends after the closing flag. The transmitter issues a TXB interrupt only after sending the programmed number of pads to the Tx FIFO buffer. The user can use the PAD value to guarantee that the TXB interrupt occurs after the closing flag has been sent out on the TXD line. PAD = 0, means that the TXB interrupt is issued immediately after the closing flag is sent to the Tx FIFO buffer. The number of PAD characters depends on the FIFO size assigned to the channel in the MCC hardware. If the channel is not part of a super channel then the MCC hardware assigns to this channel a FIFO of 4 bytes. So in this case a pad of 4 bytes ensures that the TXB interrupt is not given before the closing flag has been sent over the TXD line. For a super channel, FIFO length equals the number of channels included in the super channel multiplied by two.

The data length and buffer pointer are described below:

- Data length. The data length is the number of bytes the MCC should transmit from this BD's data buffer. It is never modified by the CP. The value of this field should be greater than zero.
- Tx buffer pointer. The transmit buffer pointer, which contains the address of the associated data buffer, may be even or odd provided that SS7_OPT[SEN_FIS] = 0 (refer to Section 44.6.4, "SS7 Configuration Register (SS7_OPT)"). If the automatic FISU option is required then the buffer pointer must be 4-byte aligned. The buffer may reside in either internal or external memory. This value is never modified by the CP.

45.9 SS7 Initialization and Start/Stop Sequence

The initialization and start/stop sequence of SS7 channel is identical to MCC initialization and start/stop sequence. See Chapter 33, "Multi-Channel Controllers (MCCs)," in the *MPC8560 User's Manual*.

Index

Numerics

60x bus
  data transfers
    port size, 12-92
  port size device interfaces, 12-93

A

Accessing dual-port RAM, 20-32
Accumulator, 6-48
Address maps
  addressing on PCI bus, 15-54
Address translation
  overview, 5-26
Addressing
  PCI bus
    addressing, 15-56
    configuration space, 15-56
    I/O space, 15-56
    memory space, 15-55
AD*n* (PCI address/data bus) signals, 15-55
Alignment, byte, 15-57
AppleTalk mode
  GSMR, 32-3
  programming example, 32-3
  PSMR, 32-4
  TODR, 32-4
Applications
  examples, 1-25
APUs, accumulator, 6-48
Arbitration
  I^2C arbitration procedure, 11-14
  PCI bus, 15-49
Architecture
  overview, 1-9
ATM
  protocol converter, 1-36
ATM controller
  AAL1 sequence number protection table, 35-81
  AALn RxBD, 35-7, 35-72
  AALn TxBD, 35-5, 35-77
  ABR flow control, 35-9, 35-21
  address compression, 35-16
  ATM layer statistics, 35-36
ATM memory structure, 35-39
ATM pace control (APC) unit
  ATM service types, 35-9
  configuration, 35-101
  data structures, 35-64
  modes, 35-9
  overview, 35-9
  parameter tables, 35-65
  priority table, 35-66
  scheduling mechanism, 35-10
  scheduling tables, 35-66
  traffic type, 35-12
  UBR+ traffic, 35-14
  VBR traffic, 35-13
atm transmit command, 35-97
ATM-to-ATM data forwarding, 35-39
ATM-to-TDM interworking, 35-36
buffer descriptors, 35-67
exceptions, 35-82
FCCE, 35-90
FCCM, 35-90
features list, 35-2
FPSMR, 35-88
GFMR register, 35-88
global mode entry (GMODE), 35-43
interrupt queues, 35-82
maximum performance configuration, 35-100
OAM performance monitoring, 35-31, 35-63
OAM support, 35-29
operations and maintenance (OAM) support, 35-29
overview, 35-5
parameter RAM, 35-40
performance monitoring, 35-9
performance, maximum (configuration), 35-100
programming model, 35-88
receive connection table (RCT)
  AALn protocol-specific RCTs, 35-48-??
  ATM channel code, 35-45
  overview, 35-44
  raw cell queue, 35-20
  RCT entry format, 35-46
registers, 35-88
RxBD, 35-72
RxBD extension, 35-76

SRTS generation using external logic, 35-98
transmit connection table (TCT)
AALn protocol-specific TCTs, 35-56
ATM channel code, 35-45
overview, 35-44
TCT entry format, 35-53
transmit connection table extension (TCTE)
ABR protocol-specific, 35-60
ATM channel code, 35-45
overview, 35-44
UBR+ protocol-specific, 35-59
VBR protocol-specific, 35-58
TxBD, 35-77
TxBD extension, 35-81
UDC extended address mode, 35-35
UEAD_OFFSET determination, 35-42
UNI statistics table, 35-82
user-defined cells (UDC)
extended address mode, 35-35
overview, 35-35
RxBD extension (AAL5/AAL1), 35-76
TxBD extension (AAL5/AAL1), 35-81
user-defined RxBD extension (AAL5/AAL1), 35-76
user-defined TxBD extension (AAL5/AAL1), 35-81
UTOPIA interface, 35-85
VCI filtering, 35-42
VCI/VPI address lookup, 35-15
VC-level address compression tables (VCLT), 35-19
VP-level address compression table (VPLT), 35-18
Atomic update memory references, 5-28

B

Baud-rate generator (BRG)
BRGCLK, 43-2
memory map, 2-50, 20-12
BDLE (SCC BISYNC DLE) register, 30-8
BISYNC mode
commands, 30-5
control character recognition, 30-6
error handling, 30-10
frame reception, 30-3
frame transmission, 30-2
frames, classes, 30-1
memory map, 30-4
overview, 30-1
parameter RAM, 30-4
programming example, 30-18
programming the controller, 30-17
receiving synchronization sequence, 30-9
RxBD, 30-12
sending synchronization sequence, 30-9
TxBD, 30-14
Block diagram, 11-1, 12-1
Block diagrams
cascaded mode, 25-4
communications processor (CP), 20-22
communications processor module (CPM), 1-14, 20-3
CPM multiplexing logic (CMX), 23-2
DMA controller, 14-1
DPLL receiver, 27-22
dual-port RAM, 20-32
Fast Ethernet, 39-3
FCC overview, 34-3
I^2C controller, 43-1
parallel I/O ports, 44-5
RapidIO interface, 16-3
SCC block diagram, 27-2
serial interface, 22-2
serial peripheral interface (SPI), 42-1
timers, 25-1
Book E
32-bit implementations, 6-3
interrupt registers, 6-16
Boot page translation, 4-20
Boot sequencer
general, 4-21
overview, 1-18
Branch target buffer (BTB)
entry address register (BBEAR), 6-23
registers, 6-23
target address register (BBTAR), 6-23
Branch unit control and status register 0 (BUCSR0), 6-23, 6-24
BRGCLK, 43-2
BSYNC (BISYNC SYNC) register, 30-7
Buffer descriptors
ATM controller
receive, 35-68, 35-72
transmit, 35-67, 35-77
BISYNC mode, 30-12
definition, 36-25
fast communications controllers (FCCs)
Fast Ethernet mode
receive, 39-25
transmit, 39-28
HDLC mode
receive, 40-9
transmit, 40-12
overview
receive, 34-10
transmit, 34-10
HDLC mode, 29-8
I^2C controller
receive, 43-13
transmit, 43-14

multi-channel controllers (MCCs)
receive, 33-33
transmit, 33-35
overview, 27-11, 36-40
serial peripheral interface (SPI)
receive, 42-15
transmit, 42-16
transparent mode
serial communications controllers (SCCs), 31-9
UART mode
serial communications controllers (SCCs), 28-15
Burst operations
ordering
PCI cache wrap mode, 15-55
PCI linear incrementing, 15-55
PCI bus transfer, 15-52
Bus operations
PCI bus transactions, *see* PCI interface
Byte
enable signals, 15-8, 15-56, 15-57
ordering, PCI bus, 15-3
PCI alignment, 15-57
Byte stuffing, 30-1
Byte, alignment, 15-57

C

$\overline{\text{C/BE}}n$ (command/byte enable) signals, 15-8, 15-57, 15-75
Cache
chance wrap mode, PCI, 15-55
control instructions, overview, 5-28
data
configuration, 7-2
organization, 7-3
L1 cache
configuration register 0, 6-29
configuration register 1, 6-30
on-chip implementation, 5-19
Cascaded mode, 25-3
CCSRBAR, updating, 4-17
CHAMR (channel mode register), 33-12
CHAMR (channel mode register, transparent mode), 33-15, 36-51
Clock glitch detection, 27-26
Clocking
boot page translation register (BPTR), 4-21
Ethernet, 4-16
introduction, 4-14
overview, 1-21
RapidIO, 4-15
real time, 4-16
system clock/PCI clock, 4-14
Clocks
clock stretching, 11-15
clock synchronization, 11-14
SCC clock glitch detection, 27-26
SCCR, 24-2
signal description, 4-2
signals *see* Signals, clock
CMXFCR (CMX FCC clock route register), 23-12
CMXSCR (CMX SCC clock route register), 23-15
CMXSI1CR (CMX SI1 clock route register), 23-11
CMXSI2CR (CMX SI2 clock route register), 23-11
CMXUAR (CMX UTOPIA address register), 23-7
Commands
atm transmit command, 35-97
fast communications controllers (FCCs)
Ethernet mode
receive commands, 39-14
transmit commands, 39-13
HDLC mode
receive commands, 40-6
transmit commands, 40-5
I^2C controller, 43-11
multi-channel controllers (MCCs)
receive commands, 33-28
PCI
$\overline{\text{C/BE}}n$ signals, 15-53
encodings, 15-53
interrupt-acknowledge transaction, 15-73
special-cycle, 15-73
serial peripheral interface (SPI), 42-13
communication processor module
features,, 38-8
Communications performance, 1-38
Communications processor (CP)
block diagram, 20-22
execution from RAM, 20-24
features list, 20-20
memory map, 2-50, 20-13
microcode execution from RAM, 20-24
microcode revision number, 20-27
peripheral interface, 20-22
RCCR, 20-24
REV_NUM, 20-27
RTSCR, 20-25
RTSR, 20-26
Communications processor module (CPM)
ATM controller
AAL1 sequence number protection table, 35-81
AALn RxBD, 35-7, 35-72
AALn TxBD, 35-5, 35-77
ABR flow control, 35-9, 35-21
address compression, 35-16
ATM layer statistics, 35-36
ATM memory structure, 35-39

ATM pace control (APC) unit
ATM service types, 35-9
configuration, 35-101
data structure, 35-64
modes, 35-9
overview, 35-9
parameter tables, 35-65
priority table, 35-66
scheduling mechanism, 35-10
scheduling tables, 35-66
traffic type, 35-12
UBR+ traffic, 35-14
VBR traffic, 35-13
atm transmit command, 35-97
ATM-to-ATM data forwarding, 35-39
ATM-to-TDM interworking, 35-36
buffer descriptors, 35-67
exceptions, 35-82
FCCE, 35-90
FCCM, 35-90
features list, 35-2
FPSMR, 35-88
GFMR register, 35-88
global mode entry (GMODE), 35-43
interrupt queues, 35-82
maximum performance configuration, 35-100
OAM performance monitoring, 35-31, 35-63
OAM support, 35-29
operations and maintenance (OAM) support, 35-29
overview, 35-5
parameter RAM, 35-40
performance monitoring, 35-9
performance, maximum (configuration), 35-100
programming model, 35-88
receive connection table (RCT)
AALn protocol-specific RCTs, 35-48-35-52
ATM channel code, 35-45
overview, 35-44
raw cell queue, 35-20
RCT entry format, 35-46
registers, 35-88
RxBD, 35-72
RxBD extension, 35-76
SRTS generation using external logic, 35-98
transmit connection table (TCT)
AALn protocol-specific TCTs, 35-56-35-58
ATM channel code, 35-45
overview, 35-44
TCT entry format, 35-53
transmit connection table extension (TCTE)
ABR protocol-specific, 35-60
ATM channel code, 35-45
overview, 35-44
UBR+ protocol-specific, 35-59
VBR protocol-specific, 35-58
TxBD, 35-77
TxBD extension, 35-81
UDC extended address mode, 35-35
UEAD_OFFSET determination, 35-42
UNI statistics table, 35-82
user-defined cells (UDC)
extended address mode, 35-35
overview, 35-35
RxBD extension (AAL5/AAL1), 35-76
TxBD extension (AAL5/AAL1), 35-81
user-defined RxBD extension (AAL5/AAL1), 35-76
user-defined TxBD extension (AAL5/AAL1), 35-81
UTOPIA interface, 35-85
VCI filtering, 35-42
VCI/VPI address lookup, 35-15
VC-level address compression tables (VCLT), 35-19
VP-level address compression table (VPLT), 35-18
block diagram, 1-14, 20-3
command set
command descriptions, 20-29
command execution latency, 20-31
command register example, 20-31
CPCR, 20-27
opcodes, 20-29
overview, 20-27
communications processor (CP)
block diagram, 20-22
execution from RAM, 20-24
features list, 20-20
microcode execution from RAM, 20-24
microcode revision number, 20-27
peripheral interface, 20-22
RCCR, 20-24
REV_NUM, 20-27
RTSCR, 20-25
RTSR, 20-26
CPM multiplexing logic (CMX)
block diagram, 23-2
overview, 23-1
dual-port RAM
accessing dual-port RAM, 20-32
block diagram, 20-32
buffer descriptors, 20-35
memory map, 20-33
overview, 20-31
parameter RAM, 20-35
fast communications controllers (FCCs)
Fast Ethernet mode

address recognition, 39-16
block diagram, 39-3
CAM interface, 39-9
collision handling, 39-19
connecting to the MPC8260, 39-5
error handling, 39-20
FCCE, 39-23
FCCM, 39-23
features list, 39-3
frame reception, 39-7
frame transmission, 39-6
hash table algorithm, 39-18
hash table effectiveness, 39-19
interpacket gap time, 39-19
interrupt events, 39-25
loopback mode, 39-19
parameter RAM, 39-10
programming model, 39-13
RMON support, 39-15
RxBD, 39-25
TxBD, 39-28
fast Ethernet mode
hash table effectiveness, 13-125
HDLC mode
bit stuffing, 40-1
error control, 40-1
error handling, 40-6
FCCE, 40-14
FCCM, 40-14
FCCS, 40-16
features list, 40-2
FPSMR, 40-8
frame reception, 40-3
frame transmission, 40-2
overview, 40-1
parameter RAM, 40-4
programming model, 40-5
receive commands, 40-6
reception errors, 40-7
RxBD, 40-9
transmission errors, 40-7
transmit commands, 40-5
TxBD, 40-12
overview
block diagram, 34-3
disabling FCCs, 34-20
FCCEx, 34-15
FCCMx, 34-15
FCCSx, 34-15
FCRx, 34-13
FDSRx, 34-8
FPSMRx, 34-8
FTODRx, 34-9
GFMRx, 34-3
initialization, 34-15
interrupt handling, 34-16
interrupts, 34-14
parameter RAM, 34-11
RxBD, 34-9
saving power, 34-22
switching protocols, 34-22
timing control, 34-17
TxBD, 34-9
transparent mode
achieving synchronization, 41-2
external synchronization signals, 41-3
features list, 41-1
in-line synchronization pattern, 41-3
receive operation, 41-2
synchronization example, 41-4
transmit operation, 41-2
features list, 20-1
I^2C controller
block diagram, 43-1
BRGCLK, 43-2
clocking and pin functions, 43-2
commands, 43-11
features list, 43-2
loopback testing, 43-4
master read (slave write), 43-4
master write (slave read), 43-4
multi-master considerations, 43-5
parameter RAM, 43-9
programming model, 43-6
registers, 43-6
RxBD, 43-13
slave read (master write), 43-4
slave write (master read), 43-4
transfers, 43-3
TxBD, 43-14
multi-channel controllers (MCCs)
CHAMR
HDLC mode, 33-12
transparent mode, 33-15, 36-51
channel extra parameters, 33-6
commands, 33-27
data structure organization, 33-3
exceptions, 33-28
features list, 33-1
global parameters, 33-5, 36-51
HDLC parameters (channel-specific), 33-10
initialization, 33-37
INTMSK, 33-11, 36-53
latency, 33-38
MCCE, 33-30
MCCFx, 33-26
MCCM, 33-30
parameters for transparent operation, 33-14

performance, 33-38
receive commands, 33-28
RSTATE, 33-13
RxBD, 33-33
super channel table, 33-7
TSTATE, 33-11
TxBD, 33-35
overview, 1-13
overview, CPM, 20-1
parallel I/O ports
block diagram, 44-5
features, 44-1
overview, 44-1
PDATx, 44-2
PDIRx, 44-3
pin assignments (port A–port D), 44-8-44-20
PODRx, 44-2
port C interrupts, 44-20
port pin functions, 44-6
PPAR, 44-4
programming options, 44-8
PSORx, 44-4
registers, 44-2
resetting registers and parameters for all channels, 20-28
RISC timer tables
CP loading tracking, 20-42
features list, 20-37
initializing RISC timer tables, 20-40
interrupt handling, 20-41
overview, 20-36
parameter RAM, 20-37
RAM usage, 20-37
RTER, 20-39
RTMR, 20-39
scan algorithm, 20-41
SET TIMER command, 20-40
table entries, 20-39
timer counts, comparing, 20-42
TM_CMD, 20-38
tracking CP loading, 20-42
SDMA channels
LDTEM, 26-3
PDTEM, 26-3
programming model, 26-2
registers, 26-2
SDMR, 26-3
SDSR, 26-2
serial configuration, 20-19
serial peripheral interface (SPI)
block diagram, 42-1
clocking and pin functions, 42-2
commands, 42-13
configuring the SPI, 42-3
features list, 42-1
interrupt handling, 42-19
master mode, 42-3
maximum receive buffer length (MRBLR), 42-12
multi-master operation, 42-4
parameter RAM, 42-11
programming example
master, 42-17
slave, 42-18
programming model, 42-7
RxBD, 42-15
slave mode, 42-4
SPCOM, 42-11
SPIE, 42-10
SPIM, 42-10
SPMODE, 42-7
TxBD, 42-16
system interface unit (SIU)
add flexibility to CPM interrupt priorities, 21-4
encoding the interrupt vector, 21-6
FCC relative priority, 21-5
flexibility of interrupt priorities, 21-4
highest priority interrupt, 21-5
interrupt controller features list, 21-1
interrupt priorities, add flexibility, 21-4
interrupt source priorities, 21-2
interrupt vector calculation, 21-6
interrupt vector encoding, 21-6
interrupt vector generation, 21-6
masking interrupt sources, 21-5
MCC relative priority, 21-5
port C interrupts, 21-8
programming model, 21-9
registers, 21-9
SCC relative priority, 21-5
SCPRR_H, 21-9
SCPRR_L, 21-10, 38-11
SICR, 21-9
SIEXR, 21-15
SIMR_H, 21-13
SIMR_L, 21-13
SIPNR_H, 21-11
SIPNR_L, 21-12
SIVEC, 21-14
timers
memory map, 2-42, 20-5
Communications protocol table, 1-37
Communications systems examples, 1-27
Compatibility issues
overview, 1-36
Completion, PCI, 15-61
Configuration
accessing registers
CONFIG_ADDR and CONFIG_DATA, 15-68

header registers
PCI registers, *see* Registers, PCI interface, 15-67
PCI configuration cycles
configuration space header, 15-66
type 0 and 1 accesses, 15-68
PCI space addressing, 15-56
status registers
processor bus error status registers, 15-74-15-76
Configuration space, accessing alternate, 4-18
Configurations
3G wireless base station, 1-32
ATM protocol converter, 1-36
cellular base station, 1-31
frame relay card, 1-35
general, 1-37
high-performance system, 1-27
LAN-to-WAN bridge router, 1-30
main device system, 1-25
pin, 1-38
regional office router, 1-29
remote access server, 1-28
SONET transmission controller, 1-34
telecommunications switch controller, 1-33
Context ID registers, 19-25-19-26
Core complex bus (CCB)
overview, 5-29
CPCR (CP command register), 20-27
CPM multiplexing logic (CMX)
overview, 23-1
see also Serial interface (SI)
CPM multiplexing, see CPM multiplexing logic (CMX)
CPM MUX memory map, 2-54, 20-17
CPM MUX, see CPM multiplexing logic (CMX)
Critical save/restore register 0, 6-17
Critical save/store register 1, 6-17
CSRR0 (critical save/restore register 0), 6-17
CSRR1 (critical save/restore register 1), 6-17
Current context ID register (CCIDR), 19-26

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, 1-16
DDR SDRAM interface debug, 19-29

DEAR, 6-17
Debug mode
block diagram, 19-1
DDR SDRAM interface, 19-4, 19-29
detailed signal descriptions, 19-6
external signal description, 19-5
features, 19-3
functional description, 19-27
information on debug Pins, 19-29
information on ECC pins, 19-30
L1_TSTCLK input signal, 19-10
L2_TSTCLK input signal, 19-10
local bus interface, 19-4, 19-30
LSSD_MODE input signal, 19-10
memory data valid (MDVAL) output, 19-6
memory ECC (MECC{0-7}) output, 19-6
memory map and register definition, 19-10
memory source ID (MSRCID{0-4}) output, 19-7
operation, 19-3
overview, 19-2
PCI address (PCI_AD{63-59}) output signal, 19-8
PCI/PCI-X
interface, 19-29
PCI/PCI-X interface, 19-4
TCK (JTAG test clock) signal, 19-8
TDI (JTAG test data input) signal, 19-9
TDO (JTAG test data output) signal, 19-9
test and configuration signals, 19-8
TEST_SEL input signal, 19-10
TMS (JTAG test mode select) signal, 19-9
trace buffer, 19-5
trigger in (TRIG_IN) input signal, 19-7
trigger out (TRIG_OUT) output signals, 19-8
watchpoint monitor, 19-4
watchpoint trigger signals, 19-7
Decrementer auto-reload register (DECAR), 6-16
descriptor queue/enqueue pointer address register (ODQEPAR), 16-74
$\overline{\text{DEVSEL}}$ (device select) signal, 15-8, 15-56
DH*n*/DL*n* (data bus) signals, 9-6
Diagram
block, 11-1, 12-1
Diagrams
debug block, 19-1
Differences between MPC8560 and MPC8260, 1-37
Digital phase-locked loop (DPLL) operation, 27-22
DMA controller
alternate mode transfer, 14-34
bandwidth control, 14-37
basic chaining mode, 14-31
basic chaining single-write start mode, 14-32
basic channel continue mode, 14-36
basic direct mode, 14-30
basic direct single-write start mode, 14-31
basic mode transfer, 14-30
block diagram, 14-1
byte count registers (BCR*n*), 14-23
chaining mode, extended, 14-33
chaining single-write start mode, extended, 14-34
channel abort, 14-36
channel continue mode, cascading transfer chains, 14-35
channel continue mode, extended, 14-36
channel operation, 14-29
channel state, 14-37
control mode transfer, external, 14-34
current link descriptor address registers (CLNDAR*n*), 14-14
current list-alternate base descriptor address registers (CJABDAR*n*), 14-24
descriptors, 14-38
destination
address registers (DAR*n*), 14-21
address registers for RapidIO maintenance writess (DARM*n*), 14-22
attributes registers (DATR*n*), 14-19
stride registers (DSR*n*), 14-26
direct mode, extended, 14-33
direct single-write start mode, extended, 14-33
errors, 14-38
features, 14-2
functional description, 14-29
general status register (DGSR), 14-27
limitations and restrictions, 14-41
memory map/register definition, 14-7
mode registers (MR*n*), 14-10
mode transfer, extended, 14-33
modes of operation, 14-2
next link descriptor address registers (NLNDAR*n*), 14-23
next list descriptor address register (NLSDAR*n*), 14-25
overview, 14-2
register descriptions, 14-10-14-29
register summary, 14-8
signal description, external, 14-5
source address registers (SAR*n*), 14-18
source address registers for RapidIO maintenance reads (SARM*n*), 14-18
source attributes registers (SATR*n*), 14-16
source stride registers (SSR*n*), 14-26
status registers (SR*n*), 14-13
stride size and distance, 14-37
system considerations, 14-42
transfer interfaces, 14-38
unusual scenarios, 14-44
Doze mode, 1-21
DSR (data synchronization register)

overview, 27-9
UART mode, 28-11
Dual-port RAM
accessing dual-port RAM, 20-32
block diagram, 20-32
buffer descriptors, 20-35
memory map, 20-33
overview, 20-31
parameter RAM, 20-35

E

e500 coherency module (ECM)
overview, 1-16
e500 core overview, 1-9
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, 15-76
transactions, 15-74
reporting
address/data parity, 15-62
PCI bus, 15-74
$\overline{\text{PERR}}$ and $\overline{\text{SERR}}$ signals, 15-76
processor bus status register, 15-74
target-initiated termination, 15-61
retry transactions, 15-61
target-abort, 15-61
target-disconnect, 15-61
ESR, 6-19
Ethernet clocks, 4-16
Ethernet mode
fast communications controller (FCC)
address recognition, 39-16
block diagram, 39-3
CAM interface, 39-9
collision handling, 39-19
connecting to the MPC8260, 39-5
error handling, 39-20
FCCE, 39-23
FCCM, 39-23
features list, 39-3
frame reception, 39-7
frame transmission, 39-6
hash table algorithm, 39-18
hash table effectiveness, 13-125, 39-19
interpacket gap time, 39-19
interrupt events, 39-25
loopback mode, 39-19
parameter RAM, 39-10
programming model, 39-13
RMON support, 39-15
RxBD, 39-25
TxBD, 39-28
Ethernet mode register,, 39-21
Exception handling
extended model, 5-19
overview, 5-31
Exceptions
syndrome register, 6-19

F

Fast communications controllers (FCCs)
Fast Ethernet mode
address recognition, 39-16
block diagram, 39-3
CAM interface, 39-9
collision handling, 39-19
connecting to the MPC8260, 39-5
error handling, 39-20
FCCE, 39-23
FCCM, 39-23
features list, 39-3
frame reception, 39-7
frame transmission, 39-6
hash table algorithm, 39-18
hash table effectiveness, 39-19
interpacket gap time, 39-19
interrupt events, 39-25
loopback mode, 39-19
parameter RAM, 39-10
programming model, 39-13
RMON support, 39-15
RxBD, 39-25
TxBD, 39-28
fast ethernet mode
hash table effectiveness, 13-125
HDLC mode
bit stuffing, 40-1
error control, 40-1

error handling, 40-6
FCCE, 40-14
FCCM, 40-14
FCCS, 40-16
features list, 40-2
FPSMR, 40-8
frame reception, 40-3
frame transmission, 40-2
overview, 40-1
parameter RAM, 40-4
programming model, 40-5
receive commands, 40-6
reception errors, 40-7
RxBD, 40-9
transmission errors, 40-7
transmit commands, 40-5
TxBD, 40-12
overview
block diagram, 34-3
disabling FCCs, 34-20
FCCEx, 34-15
FCCMx, 34-15
FCCSx, 34-15
FCRx, 34-13
FDSRx, 34-8
FPSMRx, 34-8
FTODRx, 34-9
GFMRx, 34-3
initialization, 34-15
interrupt handling, 34-16
interrupts, 34-14
parameter RAM, 34-11
RxBD, 34-9
saving power, 34-22
switching protocols, 34-22
timing control, 34-17
TxBD, 34-9
switching protocols, 34-22
transparent mode
features list, 41-1
receive operation, 41-2
synchronization
achieving, 41-2
example, 41-4
external signals, 41-3
in-line pattern, 41-3
transmit operation, 41-2
FCCE register
ATM, 35-90
Ethernet, 39-23
FCC overview, 34-15
HDLC, 40-14
FCCM register
ATM, 35-90
Ethernet, 39-23
FCC overview, 34-15
HDLC, 40-14
FCCS (FCC status) register, 34-15, 40-16
FCRx (function code registers), 34-13
FDSRx (FCC data synchronization registers), 34-8
features
communication processor module,, 38-8
Features list, 5-5
SIU interrupt controller, 21-1
Features lists
communications processor (CP), 20-20
communications processor module (CPM), 20-1
ATM controller, 35-2
parallel I/O ports, 44-1
CPM multiplexing, 23-2
fast communications controllers (FCCs)
Fast Ethernet, 39-3
HDLC mode, 40-2
transparent mode, 41-1
HDLC bus controller, 29-20
I^2C controller, 43-2
multi-channel controllers (MCCs), 33-1
RISC timer tables, 20-37
serial communications controllers (SCCs)
AppleTalk mode, 32-2
BISYNC mode, 30-2
general list, 27-2
HDLC mode, 29-2
transparent mode, 31-1
UART mode, 28-2
serial interface, 22-3
serial peripheral interface (SPI), 42-1
timers, 25-2
FPSMR register
HDLC, 40-8
protocol-specific mode, 34-8
$\overline{\text{FRAME}}$ signal, 15-52
FTODRx (FCC transmit-on-demand registers), 34-9
Full-on mode, 1-21

G

GCI
programming, 22-30
support, 22-29
General-purpose registers (GPRs), 6-8
GFMR (general FCC mode register), 34-3, 35-88
Global control register 0, 5-29
Global control register 0 (PMGC0), 6-50
Global utilities
alternate function signal multiplex control register (PMUXCR), 17-14
clock control, 17-2

clock out control register (CLKOCR), 17-21
DDR DLL control register (DDRDLLCR), 17-21
detailed signal descriptions, 17-3
device disable register (DEVDISR), 17-15
doze mode, 17-24
dynamic power management, 17-24
external signal description, 17-2
features, 17-1
functional description, 17-23
general-purpose I/O signals, 17-2
I/O control register, general-purpose (GPIOCR), 17-11
I/O signals, general-purpose, 17-27
input data register, general-purpose (GPINDR), 17-13
interrupt and local bus signal multiplexing, 17-2
interrupts and power management, 17-26
local bus DLL control register (LBDLLCR), 17-22
machine check summary register (MCPSUMR), 17-19
memory map/register definition, 17-4
nap mode, 17-25
output data register, general-purpose (GPOUTDR), 17-12
overview, 17-1
POR
boot mode status register (PORBMSR), 17-6
configuration register, general-purpose (GPPORCR), 17-11
debug mode status register (PORDBGMSR), 17-10
device status register (PORDEVSR), 17-9
I/O impedance status and control register (PORIMPSCR), 17-8
PLL status register (PORPLLSR), 17-5
power management
and device disables, 17-1
control and status register (POWMGTCSR), 17-17
control bits, 17-25
functional description, 17-23
processor version register (PVR), 17-20
register descriptions, 17-5-17-23
shutting down unused blocks, 17-24
signals overview, 17-2
sleep mode, 17-25
snooping, power-down state, 17-26
software access to current POR configuration settings, 17-1
software considerations for power management, 17-26
software-controlled power-down states, 17-24
system version register (SVR), 17-20
GMODE (global mode entry), 35-43
$\overline{\text{GNT}}$ (PCI bus grant) signal, 15-49
GSMR (general SCC mode register)
AppleTalk mode, 32-3
HDLC bus protocol, programming, 29-24
overview, 27-3
GSMR_H,, 35-92, 35-93, 35-94, 35-95
gsmr_h,, 35-92, 35-93, 35-94, 35-95

H

Hardware implementation-dependent register 0 (HID0), 6-24
Hardware implementation-dependent register 1 (HID1), 6-25
HDLC mode
accessing the bus, 29-20
bus controller, 29-18
collision detection, 29-18, 29-21
commands, 29-5
delayed RTS mode, 29-22
error handling, 29-6
fast communications controllers (FCCs)
bit stuffing, 40-1
error control, 40-1
error handling, 40-6
FCCE, 40-14
FCCM, 40-14
FCCS, 40-16
features list, 40-2
FPSMR, 40-8
frame reception, 40-3
frame transmission, 40-2
overview, 40-1
parameter RAM, 40-4
programming model, 40-5
receive commands, 40-6
reception errors, 40-7
RxBD, 40-9
transmission errors, 40-7
transmit commands, 40-5
TxBD, 40-12
features list, 29-2
GSMR, HDLC bus protocol programming, 29-24
multi-master bus configuration, 29-19
overview, 29-1
parameter RAM, 29-3
performance, increasing, 29-21
programming example, 29-16, 29-24
programming the controller, 29-5
PSMR, 29-7
RxBD, 29-8
single-master bus configuration, 29-20
TxBD, 29-12
using the TSA, 29-23

I

I2ADD (I^2C address) register, 43-7
I2BRG (I^2C baud rate generator) register, 43-7
I^2C controller
  block diagram, 43-1
  BRGCLK, 43-2
  clocking and pin functions, 43-2
  commands, 43-11
  features list, 43-2
  loopback testing, 43-4
  master read (slave write), 43-4
  master write (slave read), 43-4
  multi-master considerations, 43-5
  parameter RAM, 43-9
  programming model, 43-6
  registers, 43-6
  RxBD, 43-13
  slave read (master write), 43-4
  slave write (master read), 43-4
  transfers, 43-3
  TxBD, 43-14
I^2C 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
I2C memory map,, 2-50, 20-13
I2CER (I^2C event register), 43-8
I2CMR (I^2C mask register), 43-8
I2COM (I^2C command) register, 43-8
I2MOD (I^2C mode) register, 43-6
IDL interface programming,, 22-27
IDL interface support, 22-26
Iinterrupt and local bus signal multiplexing, 17-27
Initialization/application information, 9-54, 10-49, 12-88
Input/output port memory map, 2-41, 20-4
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, 12-57
Interrupt classes
  categories, 5-20
Interrupt registers
  defined by Book E, 6-16
  e500-specific, 6-20
  overview, 5-22
Interrupts
  and exception handling registers
    ESR, 6-19
    MSR, 6-10
  ATM interrupt queues, 35-82
  PIC
    messaging, 10-48
    nesting, 10-47
  RISC timer tables
    interrupt handling, 20-41
  SCC interrupt handling, 27-16
  vector offset registers, 6-18
  vector prefix register, 6-17
$\overline{\text{IRDY}}$ (initiator ready) signal, 15-52
IVORs, 6-18
IVPR, 6-17

J

JEDEC-standard SDRAM interface commands, 12-57
JTAG test reset ($\overline{\text{TRST}}$) signal, 19-9

K

Key features, 1-2

L

L1 cache
  configuration register 0 (L1CFG0), 6-29
  configuration register 1 (L1CFG1), 6-30
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 (L2CEWCR0–L2CEWCR03), 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 (L2SRBARn), 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, 12-60
address and address space checking, 12-38
atomic operation, 12-41
base registers (BR0–7), 12-11
boot chip-select operation, 12-54
bus clock and clock ratios, 12-4
bus monitor, 12-42
chip-select and write enable negation timing, 12-48
chip-select assertion timing, 12-47
clock ratio register (LCRR), 12-35
data buffer control (LBCTL), 12-41
data transfer acknowledge (TA), 12-40
extended hold Time on read accesses, 12-52
external access termination, 12-54
external signal description, 12-5
features, 12-2
general-purpose chip-select machine (GPCM), 12-42
local bus configuration register (LBCR), 12-34
memory data register (MDR), 12-24
memory map/register definition, 12-10
memory refresh timer prescaler register (MRTPR), 12-23
modes of operation, 12-3
option registers (OR*x*), 12-13
output enable timing, 12-52
overview, 12-2
page hit checking, 12-58
page management, 12-58
parity generation and checking (LDP), 12-42
power-down mode, 12-4
precharge-to-activate interval, 12-60
programmable wait state configuration, 12-48
register descriptions, 12-11
relaxed timing, 12-49
SDRAM machine, 12-55
SDRAM machine mode register (LSDMR), 12-25
SDRAM refresh timer (LSRT), 12-28
signal descriptions, 12-5
signals
external address latch enable (LALE), 12-38
source ID debug mode, 12-4
timing configuration, 12-43
transfer error address register (LTEAR), 12-34
transfer error attributes register (LTEATR), 12-33
transfer error check disable register (LTEDR), 12-31
transfer error interrupt enable register (LTEIR), 12-32
transfer error status register (LTESR), 12-29
unctional description, 12-37
UPM memory address register (MAR), 12-20
UPM mode registers (M*x*MR), 12-20
UPM refresh timer (LURT), 12-27
LDTEM (SDMA local bus transfer error MSNUM register), 26-3
Local bus interface debug, 19-30
Local control A registers (PMLCa0–PMLCa3), 6-51
Local control B registers (PMLCb0–PMLCb3), 6-52
Local control registers
A 0 – 3 (PMLCa0–PMLCa3), 6-51
B 0 – 3 (PMLCb0–PMLCb3), 6-52
overview, 5-30
Loopback mode, 22-7
LR, 6-10

M

MAA (memory address attribute) signals, 19-7
Machine check interrupt, 5-21
Machine state register (MSR), 6-10
Master-abort termination, PCI, 15-61
MCCE (MCC event) register, 33-30
MCCFx (MCC configuration registers), 33-26
MCCM (MCC mask) register, 33-30
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, 2-17
CCSR organization, 2-12
communications processor module and CCSR, 2-14
configuration, control, and status registers, 2-10-2-17
configuring local access windows, 2-8
device-specific utilities registers, 2-16
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
intoduction, 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
BRGs,, 20-12
PIP,, 20-8
Memory map/register definitions
L2 cache, 7-5
Memory maps
BRGs, 2-50
LBC, 12-10
PIP, 2-45
serial communications controllers (SCCs)
BISYNC mode, 30-4
HDLC mode, 29-4
Memory references, atomic update, 5-28
Memory subsystem
overview, 5-31
Microcode revision number, 20-27
MMU
assist registers (MAS0–MAS4 and MAS6), 6-34
assist registers (MAS1–MAS4 and MAS6), 5-26
registers, 6-31
TLB
coherency, 5-27
Modes
ATM controller
APC modes, 35-9
basic
chaining, 14-31
chaining single-write start, 14-32
direct, 14-30
direct single-write start, 14-31
transfer, 14-30
BISYNC mode, 30-1
cache wrap, 15-55
cascaded mode, 25-3
doze, 17-24
HDLC mode, 29-1
hunt mode, 28-10
nap, 17-25
NMSI mode, synchronization, 31-3
SCC AppleTalk mode, 32-1
serial interface (SI)
echo mode, 22-7
serial peripheral interace (SPI)
master mode, 42-3
sleep, 17-25
slow go, 25-2
transparent mode
overview, 41-1
serial communications controllers (SCCs), 31-1
UART mode
serial communications controllers (SCCs), 28-1
MPC8245
as PCI bus master, 15-4
as PCI target, 15-4
Multi-channel controllers (MCCs)
CHAMR
HDLC mode, 33-12
transparent mode, 33-15, 36-51
channel extra parameters, 33-6
commands, 33-27
data structure organization, 33-3
exceptions, 33-28
features list, 33-1
global parameters, 33-5, 36-51
HDLC parameters (channel-specific), 33-10
initialization, 33-37
INTMSK, 33-11, 36-53
latency, 33-38
MCCE, 33-30
MCCFx, 33-26
MCCM, 33-30
parameters for transparent operation, 33-14
performance, 33-38
receive commands, 33-28
RSTATE, 33-13
RxBD, 33-33

super channel table, 33-7
TSTATE, 33-11
TxBD, 33-35
Multiprocessor system
overview, 1-26

N

Nap mode description, 1-21
NMSI (non-multiplexed serial interface)
configuration, 23-4
synchronization in NMSI mode, transparent operation, 31-3

O

On-chip cache implementation, 5-19
On-chip memory
as L2 Cache, 1-15
as mapped SRAM, 1-15
unit introduction, 1-15
Operations
digital phase-locked loop (DPLL) operation, 27-22
transparent operation, NMSI sychronization, 31-3
Overflow (OV), 6-8

P

Page characteristics
e500 core, 5-28
PAR (PCI parity) signal, 15-75
Parallel I/O ports
block diagram, 44-5
features, 44-1
overview, 44-1
PDATx, 44-2
PDIRx, 44-3
pin assignments (port A–port D), 44-8-44-20
PODRx, 44-2
port C interrupts, 44-20
port pin functions, 44-6
PPAR, 44-4
programming options, 44-8
PSORx, 44-4
registers, 44-2
Parameter RAM
ATM controller, 35-40
fast communications controllers (FCCs)
Fast Ethernet mode, 39-10
HDLC mode, 40-4
overview, 34-11
HDLC mode, 29-3
I^2C controller, 43-9
serial communications controllers (SCCs)
all protocols, 27-13
base addresses, 27-15
BISYNC mode, 30-4
overview, 27-13
UART mode, 28-4
serial peripheral interface (SPI), 42-11
PCI
arbiter configuration, 4-11
debug configuration, 4-12
I/O impedence, 4-11
PCI bus
device as initiator, 15-4
device as target, 15-4
external signal descriptions, 15-6
features, 15-4
host/agent/agent lock mode, 15-5
introduction, 15-1
modes of operation, 15-5
overview, 15-2
PCI-X signal output hold timing, 15-6
PCI interface
accessing registers, see Registers, 15-69
address bus decoding, 15-55
address/data parity errors, 15-62
agent config lock mode, 15-77
burst operation, 15-52
bus arbitration, 15-49
bus commands, 15-53
bus protocol, 15-52
bus transactions
interrupt-acknowledge transaction, 15-73
read transactions, 15-58
special-cycle transaction, 15-73
timing diagrams, 15-57
transaction termination, 15-60
master-initiated, 15-60
target-initiated, 15-61
write transactions, 15-59
bus transactions, *see* PCI interface
byte alignment, 15-57
byte ordering, 15-3
$\overline{\text{C/BE}}$ signals, 15-75
$\overline{\text{C/BE}}n$ signals, 15-75
cache wrap mode, 15-55
command register, 15-67
configuration cycles
configuration header, 15-67
type 0 and 1 accesses, 15-68
configuration space addressing, 15-56
data transfers, 15-52, 15-57
error detection and reporting, 15-74
error transactions, 15-74
fast back-to-back transactions, 15-64
host mode, 15-77
I/O space addressing, 15-56

linear incrementing, 15-55
master-initiated transaction termination, 15-61
memory space addressing, 15-55
MPC8245 as PCI bus master, 15-4
MPC8245 as PCI target, 15-4
PCI commands
C/BEn signals, 15-53
command summary, 15-53
encodings, 15-53
interrupt-acknowledge transaction, 15-73
special-cycle, 15-73
power-on reset configuration modes, 15-76
read transactions, 15-58
registers, *see* Registers, PCI interface, 15-67
retry PCI transactions, 15-62
signals
DEVSEL, 15-8
TRDY, 15-13
signals, *see* Signals, PCI interface
special cycle command, 15-74
special-cycle operations, 15-73, 15-74
target-abort error, 15-62
target-disconnect, 15-4, 15-61
target-disconnect termination, 15-61
target-initiated termination, 15-62
transaction termination, 15-57, 15-60
turnaround cycle, 15-57
write transactions, 15-59
PCI/PCI-X
and RapidIO, bridging between, 15-78
inbound ATMU windows, 2-10
output hold configuration, 4-13
PCI/PCI-X interface debug, 19-29
PCI-X
configuration, 4-12
nonposted writes, 15-77
PDATx (port data) registers, 44-2
PDIRx (port data direction registers), 44-3
PDTEM (SDMA 60x bus transfer error MSNUM register), 26-3
Performance monitor
burstiness, 18-15
control registers, 18-4-18-10
counter
chaining, 18-14
registers, 18-10
triggering, 18-14
counters (PMC*n*), 18-10
event counting, 18-12
events, 18-17-18-38
examples, 18-38
external signals, 18-3
features, 18-2
functional description, 18-12
global control register (PMGC0), 18-5
interrupt, 18-12
local control registers (PMLCA*n* and PMLCB*n*), 18-5
mask registers
even (PMMR0,2,4,6), 10-29
odd (PMMR1,3,5,7), 10-30
memory map and register definition, 18-3
overview, 18-1
purposes, 5-5
threshold events, 18-13
Performance monitor APU
counter registers, 5-30, 6-53
Performance monitoring, overview, 5-29
PERR (PCI parity error) signal, 15-76
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
$\overline{\text{IRQ_OUT}}$ 0 (IRQSR0), 10-26
$\overline{\text{IRQ_OUT}}$ 1 (IRQSR1), 10-27
timers, 10-48
PIR, 6-13, 6-31
PODRx (port open-drain registers), 44-2
Power consumption
FCCs, 34-22
SCCs, 27-28
Power management
global utilities, 17-23
nap mode, 15-74
overview, 1-21
PCI special-cycle operations, 15-73
Power-on reset
output signal state, 3-16
PowerPC architecture legacy support overview, 5-30
PPAR (port pin assignment register), 44-4
Process ID registers (PID)
overview, 5-27
Processor ID register (PIR), 6-31
Programmable interrupt controller (PIC)
overview, 1-17
Programmable page characteristics, 5-28
Programmed context ID register (PCIDR), 19-25
Programming examples
serial communications controllers (SCCs)
GSMR (general SCC mode register)
AppleTalk mode, 32-3
HDLC bus protocol, 29-24
PSMR (protocol-specific mode register)
AppleTalk mode, 32-4
TODR (transmit-on-demand register)
AppleTalk mode, 32-4
transparent mode, 31-14
UART mode, 28-23
Programming model
e500 core, 5-18, 6-2
overview, 5-17
Promiscuous mode, see Transparent mode
Promiscuous operation, 41-1
PSMR (protocol-specific mode register)
AppleTalk mode, 32-4
BISYNC mode, 30-10
HDLC bus protocol, programming, 29-24
HDLC mode, 29-7
overview, 27-9
transparent mode, 31-9
UART mode, 28-14
PSMR,, 39-22
PSMR, Ethernet mode register,, 39-21
PSORx (port special options registers), 44-4
PVR, 6-13

R

RapidIO
and CCSR, 2-15
and PCI/PCI-X, bridging between, 15-78
clocks, 4-15
device ID, 4-10
inbound ATMU windows, 2-9
transmit clock source, 4-10
RapidIO interconnect
overview, 1-20
RapidIO interface
8/16 LP-LVDS layer functionality lists, 16-95
8/16 LP-LVDS port maintenance block header 0 command and status register (PMBH0CSR), 16-26
accept-all mode, 16-6
architectural registers, 16-14-16-33
assembly identity capability register (AIDCAR), 16-15
assembly information capability register (AICAR), 16-16
ATMU (address translation and mapping unit), 16-112
ATMU registers, 16-36-16-44

base device ID command and status register (BDIDCSR), 16-24
basic functionality list, 16-95
block diagram, 16-3
common transport layer
basic fFunctionality list, 16-103
detectable errors list, 16-104
packet transmission list, 16-103
common transport layer packet reception list, 16-103
component tag command and status register (CTCSR), 16-25
configuration register (CR), 16-33
CRC checking modes, 16-5
data message controller, 16-117
data message controller limitations and restrictions, 16-125
destination operations capability register (DOCAR), 16-19
detailed signal descriptions, 16-8
device identity capability register (DIDCAR), 16-14
device information capability register (DICAR), 16-14
doorbell
message controller, 16-125
mode register (DMR), 16-80
queue/dequeue pointer address registers (DQDPAR), 16-83
queue/enqueue pointer address registers (DQEPAR), 16-84
registers, 16-80-16-85
status register (DSR), 16-82
error checking disable mode, 16-6
error management registers, 16-44-16-66
errors, 16-106
external signal description, 16-6
features, 16-4
features not implemented, 16-5
functional description, 16-88
functionality, 16-94
general device
link maintenance list, 16-97
nonrecoverable errors list, 16-102
packet reception list, 16-98
packet transmission list, 16-97
recoverable errors list, 16-100
general specification functionality list, 16-94
host base device ID lock command and status register (HBDIDLCSR), 16-25
inbound
frame queue/dequeue pointer address register (IFQDPAR), 16-79
frame queue/enqueue pointer address register (IFQEPAR), 16-79
message registers, 16-75-16-85
mode register (IMR), 16-75
status register (ISR), 16-77
window attributes registers 0–4 (RIWAR*n*), 16-42
window base address registers 1–4 (RIWBAR*n*), 16-42
window translation address registers 0–4 (RIWTAR*n*), 16-41
inbound ATMU
bypass mode, 16-114
LCSRBAR window, 16-114
translation, 16-114
inbound special transactions and requirements, 16-115
initialization and application information, 16-129
introduction, 16-1
link response time-out disable mode, 16-6
local configuration space base address 1 command and status register (LCSBA1CSR), 16-23
logical layer
basic functionality list, 16-104
detectable errors list, 16-106
source transaction support list, 16-106
target transaction support list, 16-105
mailbox command and status register (MSR), 16-21
memory map/register definition, 16-9
message unit, 16-116
chaining mode operation, 16-118
description, 16-117
descriptor format, 16-122
direct mode operation, 16-118
doorbell controller interrupts, 16-127
doorbell queue entry format, 16-127
error response conditions, 16-125
features, 16-116
inbound doorbell reception, 16-126
inbox controller interrupts, 16-125
inbox controller operation, 16-123
operational modes, 16-117
outbox controller operation, 16-118
port-write controller structure, 16-128
response conditions, 16-127
retry response conditions, 16-125
special error case condition, 16-123
message unit registers, 16-67
operational modes, 16-5
outbound, 16-74
descriptor queue/dequeue pointer address register (ODQDPAR), 16-70
destination attributes register (ODATR), 16-72
destination port register (ODPR), 16-72
double-word count register (ODCR), 16-73
message registers, 16-67-16-75
mode register (OMR), 16-67

special transactions and requirements, 16-113
status register (OSR), 16-69
unit source address register (OSAR), 16-71
outbound ATMU bypass mode, 16-113
outbound ATMU translation, 16-113
outbound window
attributes registers 0–8 (ROWAR*n*), 16-39
base address registers 1–8 (ROWBAR*n*), 16-38
translation address registers 0–8 (ROWTAR*n*), 16-36
translation address registers 0–8 for maintenance transactions (ROWTAR*n*), 16-38
output port driver disable mode, 16-6
overview, 16-7
packet error capture register 1
type 1 (PECR1T1), 16-52
type 10 request (PECR1T10), 16-56
type 11 (PECR1T11), 16-57
type 13 (PECR1T13), 16-57
type 2 (PECR1T2), 16-53
type 5 (PECR1T5), 16-53
type 6 (PECR1T6), 16-54
type 8 request (PECR1T8RQ), 16-55
type 8 response (PECR1T8R), 16-55
packet error capture register 2
type 10 (PECR2T10), 16-61
type 6 (PECR2T6), 16-59
type 8 (PECR2T8), 16-60
type 8 response (PECR2T8R), 16-60
types 1, 2, and 5 (PECR1T125), 16-58
types 11 and 13 (PECR2T11), 16-62
packet response time-out disable mode, 16-6
port configuration register (PCR), 16-34
port control command and status register (PCCSR), 16-32
port error
and status command and status register (PESCSR), 16-31
control register (PECR), 16-50
injection register (PEIR), 16-35
packet register 1 (PEPR1), 16-52
packet register 2 (PEPR2), 16-58
packet/control symbol register 0 (PEPCSR0), 16-51
recovery threshold register (PERTR), 16-65
port general control command and status register (PGCCSR), 16-28
port link
maintenance request command and status register (PLMREQCSR), 16-29
maintenance response command and status register (PLMRESPCSR), 16-29
time-out control command and status register (PLTOCCSR), 16-27
port local ackID status command and status register (PLASCSR), 16-30
port notification/fatal error
detect register (PNFEDR), 16-44
disable register (PNFEDiR), 16-47
interrupt enable register (PNFEIER), 16-49
port recoverable error detect register (PREDR), 16-62
port response time-out control command and status register (PRTOCCSR), 16-27
port retry threshold register (PRTR), 16-66
port-write
and doorbell command and status register (PWDCSR), 16-22
mode register (PWMR), 16-85
queue base address register (PWQBAR), 16-87
registers, 16-85-16-88
status register (PWSR), 16-86
processing element features capability Register (PEFCAR), 16-16
processing element logic layer control command and status register (PELLCCSR), 16-23
source operations capability register (SOCAR), 16-18
switch port information capability register (SPICAR), 16-17
terminology, 16-2
transaction, packet, and control symbols, 16-88
transmit clock-select mode, 16-5
RapidIO registers, 16-9-16-14
RCCR (RISC controller configuration register), 20-24
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, 17-14
AppleTalk mode
GSMR, 32-3
PSMR, 32-4
TODR, 32-4
ATM controller
FCCE, 35-90
FCCM, 35-90
FPSMR (FCC protocol-specific mode register, 35-88
GFMR register, 35-88

BISYNC mode
BDLE, 30-8
BSYNC, 30-7
PSMR, 30-10
SCCE, 30-15
SCCM, 30-15
SCCS, 30-16
branch buffer entry address, 6-23
branch buffer target address (BBTAR), 6-23
branch operations, 6-8
branch target buffer (BTB), 6-23
branch unit control and status 0 (BUCSR0), 6-23, 6-24
byte count, 14-23
chip select configuration, 9-12
chip select memory bounds, 9-11
clock out control, 17-21
communications processor (CP)
RCCR, 20-24
RTSCR, 20-25
RTSR, 20-26
communications processor module (CPM)
CPCR, 20-27
parallel I/O ports
PDATx, 44-2
PDIRx, 44-3
PODRx, 44-2
PPAR, 44-4
PSORx, 44-4
condition, 6-9
configuration, 15-67
configuration control and status register base address (CCSRBAR), 4-18
configuration header summary, 15-67
count (CTR), 6-10
CPM multiplexing
CMXFCR, 23-12
CMXSCR, 23-15
CMXSI1CR, 23-11
CMXSI2CR, 23-11
CMXUAR, 23-7
critical save/restore 0 (CSRR0), 6-17
critical save/restore1 (CSRR1), 6-17
current context ID, 19-26
current link descriptor address, 14-14
current list-alternate base descriptor address, 14-24
data exception address (DEAR), 6-17
DDR DLL control, 17-21
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), 6-16
decrementer auto-reload (DECAR), 6-16
destination address for RapidIO maintenance writes, 14-22
destination attributes, 14-19
destination stride, 14-26
device disable, 17-15
DMA destination address, 14-21
DMA register summary, 14-8
DSR
overview, 27-9
UART mode, 28-11
e500 core assist, 5-26
ECM
error address capture, 8-8
error attributes capture, 8-6
error detect, 8-5
error enable, 8-5
exception syndrome (ESR), 6-19
external interrupt vector/priority, 10-33
fast communications controller (FCC)
FCCE, 40-14
FCCS, 40-16
FPSMR, 40-8
fast communications controllers (FCCs)
Fast Ethernet mode
FCCE, 39-23
FCCM, 39-23
HDLC mode
FCCM, 40-14
overview
FCCEx, 34-15
FCCMx, 34-15
FCCSx, 34-15
FCRx, 34-13
FDSRx, 34-8
FPSMRx, 34-8
FTODRx, 34-9
GFMRx, 34-3
interrupts, 34-14
timing control, 34-17
general status, 14-27
general-purpose, 6-8
I/O control, 17-11
input data, 17-13
output data, 17-12
POR configuration, 17-11
global control, 5-29
GSMR
AppleTalk mode, 32-3
overview, 27-3
hardware implementation-dependent
0 (HID0), 6-24
1 (HID1), 6-25
HDLC mode

PSMR, 29-7
SCCE, 29-13
SCCM, 29-13
SCCS, 29-15
I^2C controller
I2ADD, 43-7
I2BRG, 43-7
I2CER, 43-8
I2CMR, 43-8
I2COM, 43-8
I2MOD, 43-6
integer exception (XER), 6-8
interrupt
defined by Book E, 6-16
general, 6-16
vector offset (IVORs), 6-18
vector prefix (IVPR), 6-17
interrupt destination, 10-34
L1 cache configuration 0 (L1CFG0), 6-29, 6-30
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, 12-11
clock ratio, 12-35
local bus configuration, 12-34
memory data, 12-24
memory refresh timer prescaler, 12-23
option, 12-13
SDRAM machine mode register, 12-25
transfer error address, 12-34
transfer error attributes, 12-33
transfer error check disable, 12-31
transfer error interrupt enable, 12-32
transfer error status, 12-29
UPM memory address, 12-20
UPM mode, 12-20
link (LR), 6-10
local bus control, 17-22
local control, 5-30
local control A (PMLCa0–PMLCa3), 6-51
local control B (PMLCb0–PMLCb3), 6-52
machine check save/store 1 (MCSRR1), 6-20
machine check summary, 17-19
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
communications processor module and CCSR, 2-14
device-specific utilities, 2-16
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, 6-31
mode, 14-10
multi-channel controllers (MCCs)
CHAMR, 33-12
CHAMR, transparent mode, 33-15, 36-51
MCCE, 33-30
MCCFx, 33-26
MCCM, 33-30
RSTATE, 33-13
TSTATE, 33-11
next link descriptor address, 14-23
next list descriptor address, 14-25
PCI interface
CONFIG_DATA and CONFIG_ADDR, 15-68
status, 15-67-15-75
performance monitor
counters, 18-10
global control, 18-5
local control, 18-5
performance monitor counter, 5-30, 6-53
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, 17-6
debug mode status, 17-10
device status, 17-9
I/O impedence status and control, 17-8
PLL status, 17-5
power management control and status, 17-17
process ID, 5-27
processor
control, 6-10
ID (PIR), 6-13, 6-31
version (PVR), 6-13
processor version, 17-20
programmed context ID, 19-25
programming PIC, 10-49
PSMR
AppleTalk mode, 32-4
BISYNC mode, 30-10
overview, 27-9
transparent mode, 31-9
UART mode, 28-14
RapidIO interface
8/16 LP-LVDS port maintenance block header 0 command and status, 16-26
assembly identity capability, 16-15
assembly information capability, 16-16
base ID command and status, 16-24
component tag command and status, 16-25
configuration, 16-33
destination operations capability, 16-19
device identity capability, 16-14
device information capability, 16-14
doorbell
mode, 16-80
queue/dequeue pointer address, 16-83
queue/enqueue pointer address, 16-84
status, 16-82
host base ID lock command and status, 16-25
inbound
frame queue/dequeue pointer address, 16-79
frame queue/enqueue pointer address, 16-79
mode, 16-75
status, 16-77
window attributes 0-4, 16-42
window base address 1-4, 16-42
window translation address 0-4, 16-41
local configuration space base address 1 command and status, 16-23
mailbox command and status, 16-21
outbound
descriptor queue/dequeue pointer address, 16-70
descriptor queue/enqueue pointer address, 16-74
destination attributes, 16-72
destination port, 16-72
double-word count, 16-73
mode, 16-67
status, 16-69
unit source address, 16-71
outbound window
attributes 0-8, 16-39
base address 1-8, 16-38
translation address, 16-36
translation address 0-8, 16-38
packet error capture 1
type 1, 16-52
type 10, 16-56
type 11, 16-57
type 13, 16-57
type 2, 16-53
type 5, 16-53
type 6, 16-54
type 8 request, 16-55
type 8 response, 16-55
packet error capture 2
type 10, 16-61
type 6, 16-59
type 8, 16-60
type 8 response, 16-60
types 1, 2, and 5, 16-58
types 11 and 13, 16-62
port configuration, 16-34
port control command and status, 16-32
port error
and status command, 16-31
control, 16-50
injection, 16-35
packet 1, 16-52
packet 2, 16-58
packet/control symbol 0, 16-51
recovery threshold, 16-65
port general control command and status register, 16-28

port link
maintenance request command and status, 16-29
maintenance response command and status, 16-29
time-out control command and status, 16-27
port local ackID status command and status, 16-30
port notification/fatal error
detect, 16-44
disable, 16-47
interrupt enable, 16-49
port recoverable error detect, 16-62
port response time-out control command and status, 16-27
port retry threshold, 16-66
port-write
and doorbell command and status, 16-22
mode, 16-85
queue base address, 16-87
status, 16-86
processing element features capability, 16-16
processing element logic layer control command and status, 16-23
source operations capability, 16-18
switch port information capability, 16-17
RFCR, 27-15
RISC timer tables
RTER, 20-39
RTMR, 20-39
TM_CMD, 20-38
save/restore 0 (SRR0), 6-16
save/restore 1 (SRR1), 6-16
SCCE
BISYNC mode, 30-15
transparent mode, 31-12
UART mode, 28-20
SCCM
BISYNC mode, 30-15
transparent mode, 31-12
UART mode, 28-20
SCCS
BISYNC mode, 30-16
transparent mode, 31-13
UART mode, 28-22
SDMA channels
LDTEM, 26-3
PDTEM, 26-3
SDMR, 26-3
SDSR, 26-2
serial interface (SI)
SIxCMDR, 22-25
SIxGMR, 22-18
SIxMR, 22-18
SIxRSR, 22-24
SIxSTR, 22-26
serial peripheral interface (SPI)
SPCOM, 42-11
SPIE, 42-10
SPIM, 42-10
SPMODE, 42-7
single-bit ECC memory error management, 9-29
software-use SPRs, USPRG0, 6-22
source address, 14-18
source address for RapidIO maintenance reads, 14-18
source attribute, 14-16
source stride, 14-26
special-purpose (SPRs)
general, 6-4
SPRG0-SPRG7, 6-22
status, 14-13
system interface unit (SIU)
SCPRR_H, 21-9
SCPRR_L, 21-10, 38-11
SICR, 21-9
SIEXR, 21-15
SIMR_H, 21-13
SIMR_L, 21-13
SIPNR_H, 21-11
SIPNR_L, 21-12
SIVEC, 21-14
system version, 17-20
TFCR, 27-15
three-speed Ether?net controller
transmit
FCS error counter, 13-86
three-speed Ethernet con?troller
jabber frame counter, 13-86
three-speed Ethernet control?ler
hash function, 13-95
Three-speed Ethernet controller
MII
management status, 13-57
transmit
control, 13-34
transmit and receive 65- and 127-byte frame counter, 13-62
three-speed Ethernet controller
AN advertisement, 13-101
AN link partner ability next page, 13-106
AN link partner base page ability, 13-103
AN next page transmit, 13-105
attribute, 13-96
attribute extract length and extract index, 13-97
carry mask register one, 13-92
carry mask register two, 13-93
carry register one, 13-90

carry register two, 13-91
control, 13-99
current receive buffer descriptor pointer, 13-43
current transmit buffer descriptor, 13-36
DMA control, 13-29
drop frame counter, 13-85
error disabled, 13-25
Ethernet control, 13-26
excessive collision packet counter, 13-84
extended status, 13-107
FIFO transmit starve, 13-33
FIFO transmit starve shutoff, 13-33
FIFO transmit threshold, 13-32
group address, 13-95
half-duplex, 13-52
individual address, 13-95
interface status, 13-58
inter-packet gap/inter-frame gap, 13-51
interrupt event, 13-21
interrupt mask, 13-24
jitter diagnostics, 13-108
MAC configuration 1, 13-48
MAC configuration 2, 13-50
maximum frame length, 13-54
maximum receive buffer length, 13-43
MII
  management address, 13-56
  management control, 13-57
  management indicator, 13-58
MII management command, 13-55
MII management configuration, 13-54
minimum frame length, 13-28
pause time value, 13-28
receive
  alignment error counter, 13-71
  broadcast packet counter, 13-68
  byte counter, 13-66
  carrier sense error counter, 13-73
  code error counter, 13-72
  control, 13-41
  control frame packet counter, 13-69
  descriptor base address, 13-45
  dropped packet counter, 13-76
  fragments counter, 13-75
  frame length error counter, 13-72
  jabber counter, 13-76
  multicast packet counter, 13-68
  oversize packet counter, 13-74
  packet counter, 13-66
  pause frame packet counter, 13-70
  status, 13-41
  undersize packet counter, 13-74
  unknown OPcode packet counter, 13-70
RxBD data length, 13-42
station address part 1, 13-59
station address part 2, 13-60
status, 13-100
TBI control, 13-109
TBI physical address, 13-31
transmit
  broadcast packet counter, 13-79
  byte counter, 13-77
  control frame counter, 13-87
  deferral packet counter, 13-80
  descriptor base, 13-37
  excessive deferral packet counter, 13-81
  fragment counter, 13-89
  multicast packet counter, 13-78
  oversize frame counter, 13-88
  packet counter, 13-78
  pause control frame counter, 13-80
  single collision packet counter, 13-82
  status, 13-35
  undersize frame counter, 13-88
transmit and receive
  1024- to 1518-byte frame counter, 13-64
  128- to 255-byte frame counter, 13-62
  1519- to 1522-byte VLAN frame counter, 13-65
  256- to 511-byte frame counter, 13-63
  512- to 1023-byte frame counter, 13-64
  64-byte frame counter, 13-61
transmit total
  collision counter, 13-84
TSEC FIFO control and status, 13-31
TxBD data length, 13-36
timer
  control (TCR), 6-13
  general, 6-13
  status (TSR), 6-14
timers
  TCN, 25-8
  TCR, 25-8
  TER, 25-8
  TGCR, 25-4
  TMR, 25-6
  TRR, 25-7
TODR
  AppleTalk mode, 32-4
  overview, 27-10
TOSEQ, 28-10
transparent mode
  PSMR, 31-9
  SCCE, 31-12
  SCCM, 31-12
  SCCS, 31-13
trigger out source, 19-27

UART mode
DSR, 28-11
PSMR, 28-14
SCCE, 28-20
SCCM, 28-20
SCCS, 28-22
TOSEQ, 28-10
user global control 0 (UPMG0), 6-51
user local control A (UPMLCa0–UPMLCa3), 6-52
user local control B (UPMLCb0–UPMLCb3), 6-53
watchpoint address, 19-13
watchpoint control 1, 19-12
watchpoint monitor
address mask, 19-14
control 0, 19-11
status, 19-17
trace buffer control 0, 19-18
transaction mask, 19-15
watchpoint monitor trace buffer
access control, 19-23
access data, 19-25
access data high, 19-24
address, 19-20
control 1, 19-20
mask, 19-21
status, 19-22
transaction, 19-22
registers
Ethernet mode,, 39-21
$\overline{\text{REQ}}$ (PCI bus request) signal, 15-49
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
receiver reset sequence, SCC, 27-27
resetting registers and parameters for all channels, 20-28
soft, 4-3
transmitter reset sequence, SCC, 27-27
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
RFCR (Rx buffer function code register)
overview, 27-15
RISC microcontroller, seeCommunications processor (CP)
RISC timer tables
CP loading tracking, 20-42
features list, 20-37
initializing RISC timer tables, 20-40
interrupt handling, 20-41
overview, 20-36
parameter RAM, 20-37
RAM usage, 20-37
RTER, 20-39
RTMR, 20-39
scan algorithm, 20-41
SET TIMER command, 20-40
table entries, 20-39
timer counts, comparing, 20-42
TM_CMD, 20-38
tracking CP loading, 20-42
RSTATE (internal receiver state) register, 33-13
RTER (RISC timer event register), 20-39
RTMR (RISC timer mask register), 20-39
RTSCR (RISC time-stamp control register), 20-25
RTSR (RISC time-stamp register), 20-26

S

Save/restore register 1 (SRR1), 6-16
SCC memory map, 2-51, 20-14
SCCE (SCC event) register
BISYNC mode, 30-15
HDLC mode, 29-13
transparent mode, 31-12
UART mode, 28-20
SCCM (SCC mask) register
BISYNC mode, 30-15
HDLC mode, 29-13
transparent mode, 31-12

UART mode, 28-20
SCCS (SCC status) register
BISYNC mode, 30-16
HDLC mode, 29-15
transparent mode, 31-13
UART mode, 28-22
SCIT programming, 22-31
SCPRR_H (CPM high interrupt priority register), 21-9
SCPRR_L (CPM low interrupt priority register), 21-10, 38-11
SDMA channels
LDTEM, 26-3
PDTEM, 26-3
programming model, 26-2
registers, 26-2
SDMR, 26-3
SDSR, 26-2
SDMR (SDMA mask register), 26-3
SDRAM
address multiplexing, 12-58
device-specific parameters, 12-59
machine, 12-55
power-on initialization, 12-56
supported configurations, 12-55
SDRAM interface
registered DIMM mode, 9-44
timing, 9-39
SDSR (SDMA status register), 26-2
Serial communications controllers (SCCs)
AppleTalk mode
connecting to AppleTalk, 32-3
operating LocalTalk frame, 32-1
overview, 32-1
programming example, 29-24, 32-4
programming the controller, 32-3
BISYNC mode
commands, 30-5
control character recognition, 30-6
error handling, 30-10
frame reception, 30-3
frame transmission, 30-2
frames, classes, 30-1
memory map, 30-4
overview, 30-1
parameter RAM, 30-4
programming example, 30-18
programming the controller, 30-17
receiving synchronization sequence, 30-9
RxBD, 30-12
sending synchronization sequence, 30-9
TxBD, 30-14
HDLC mode
accessing the bus, 29-20
bus controller, 29-18
collision detection, 29-18, 29-21
commands, 29-5
delayed RTS mode, 29-22
error handling, 29-6
features list, 29-2
GSMR, HDLC bus protocol programming, 29-24
interrupts, 29-14
memory map, 29-4
multi-master bus configuration, 29-19
overview, 29-1
parameter RAM, 29-3
performance, increasing, 29-21
programming example, 29-16, 29-24
programming the controller, 29-5
PSMR, 29-7
RxBD, 29-8
single-master bus configuration, 29-20
TxBD, 29-12
using the TSA, 29-23
overview
buffer descriptors, 27-11
controlling SCC timing, 27-18
DPLL operation, 27-22
features, 27-2
initialization, 27-17
interrupt handling, 27-16
parameter RAM, 27-13
reconfiguration, 27-26
reset sequence, 27-27
switching protocols, 27-27
transparent mode
achieving synchronization, 31-3
commands, 31-7
DSR receiver SYNC pattern lengths, 31-3
end of frame detection, 31-6
error handling, 31-8
frame reception, 31-3
frame transmission, 31-2
inherent synchronization, 31-6
in-line synchronization, 31-6
overview, 31-1
programming example, 31-14
RxBD, 31-9
synchronization signals, 31-4
synchronization, user-controlled, 31-5
transmit synchronization, 31-3
TxBD, 31-11
UART mode
commands, 28-6
control character insertion, 28-10
data handling, character and message-based, 28-5
error reporting, 28-6
features list, 28-2
fractional stop bits, 28-11

handling errors, 28-12
hunt mode, 28-10
normal asynchronous mode, 28-3
overview, 28-1
parameter RAM, 28-4
programming example, 28-23
RxBD, 28-15
S-records loader application, 28-24
status reporting, 28-6
synchronous mode, 28-3
TxBD, 28-19
Serial configuration, 20-19
Serial interface (SI)
block diagram, 22-2
enabling connections, 22-7
features, 22-3
GCI support, 22-29
IDL bus implementation
programming the IDL, 22-27
IDL interface support, 22-26
overview, 22-4
programming GCI, 22-30
programming RAM entries, 22-10
registers, 22-18
see also CPM multiplexing logic (CMX)
SI RAM, 22-8
Serial peripheral interface (SPI)
block diagram, 42-1
clocking and pin functions, 42-2
commands, 42-13
configuring the SPI, 42-3
features list, 42-1
interrupt handling, 42-19
master mode, 42-3
maximum receive buffer length (MRBLR), 42-12
multi-master operation, 42-4
parameter RAM, 42-11
programming example
master, 42-17
slave, 42-18
programming model, 42-7
RxBD, 42-15
slave mode, 42-4
SPCOM, 42-11
SPIE, 42-10
SPIM, 42-10
SPMODE, 42-7
TxBD, 42-16
$\overline{\text{SERR}}$ (system error) signal, 15-76
SI memory map, 2-54, 20-17
SI RAM programming example, 22-13
SICR (SIU interrupt configuration register), 21-9
SICR,, 23-7
SIEXR (SIU external interrupt control register), 21-15
Signals
60x processor interface DH*n*/DLn, 9-6
BIDI, 13-12
byte enable signals, 15-56
$\overline{\text{C/BE}}$*n*, 15-8, 15-57
clock, 4-2
configuration, sampled at reset, 3-14
control, description, 4-1
description, external, 14-5
detailed global utilities descriptions, 17-3
$\overline{\text{DEVSEL}}$, 15-8
EC_GTX_CLK125, 13-12
EC_MDC, 13-12
EC_MDIO, 13-12
eset and clocking blocks, 4-1
general-purpose I/O, 17-2, 17-27
global utilities external summary, 17-2
groupings, 3-3
I^2C interface
summary, 11-4
interrupt and local bus multiplexing, 17-2
introduction, 3-1
L1_TSTCLK input, 19-10
L2_TSTCLK input, 19-10
LBC external address latch enable, 12-38
local bus multiplexing, 17-27
LSSD_MODE, 19-10
MAA, 19-7
output signal states at power-on reset, 3-16
overview, 3-1
PCI address output, 19-8
PCI interface
AD*n*, 15-55
$\overline{\text{C/BE}}$*n*, 15-75
$\overline{\text{DEVSEL}}$, 15-56
$\overline{\text{FRAME}}$, 15-52
$\overline{\text{GNT}}$, 15-49
$\overline{\text{IRDY}}$, 15-52
PAR (PCI parity), 15-75
$\overline{\text{PERR}}$, 15-76
$\overline{\text{REQ}}$, 15-49
$\overline{\text{SERR}}$, 15-76
$\overline{\text{STOP}}$, 15-57
$\overline{\text{TRDY}}$, 15-53
turnaround cycle, 15-57
PIC overview, 10-7
RapidIO interface, detailed descriptions, 16-8
reset and clocking blocks, system, 4-1
system control, description, 4-2
test and configuration, 19-8
description, 19-8
TCK (JTAG test clock), 19-8
TDI (JTAG test data input), 19-9
TDO (JTAG test data output), 19-9

TMS (JTAG test mode select), 19-9
TRST (JTAG test reset), 19-9
TEST_SEL input, 19-10
TRDY, 15-13
trigger in, 19-7
trigger out, 19-8
TSEC*n*_CRS, 13-11
TSEC*n*_GTX_CLK, 13-11
TSEC*n*_RX_CLK, 13-12
TSEC*n*_RX_DV, 13-12
TSEC*n*_RX_ER, 13-13
TSEC*n*_RXD, 13-12
TSEC*n*_TX_CLK, 13-13
TSEC*n*_TX_EN, 13-14
TSEC*n*_TXD, 13-13
TSEC*n*TX_ER, 13-14
Single-processor system
overview, 1-25
SIPMR_H (SIU high interrupt mask register), 21-13
SIPMR_L (SIU low interrupt mask register), 21-13
SIPNR_H (SIU high interrupt pending register), 21-11
SIPNR_L (SIU low interrupt pending register), 21-12
SIU memory map,, 2-40, 20-3, 20-4
SIVEC (SIU interrupt vector register), 21-14
Sleep mode description, 1-21
SPCOM (SPI command) register, 42-11
SPI memory map, 2-54, 20-17
SPIE (SPI event) register, 42-10
SPIM (SPI mask) register, 42-10
SPMODE (SPI mode) register, 42-7
SPRG0-SPRG7, 6-22
SPRs, 6-4
SRAM
coherency, 7-22
features, 7-2
memory-mapped windows, 2-4
SRR0, 6-16
SRR1, 6-16
Status register, PCI, 15-57, 15-67, 15-75
STOP signal, 15-57
Summary overflow (SO), 6-8
System interface unit (SIU)
encoding the interrupt vector, 21-6
FCC relative priority, 21-5
highest priority interrupt, 21-5
interrupt controller features list, 21-1
interrupt source priorities, 21-2
interrupt vector calculation, 21-6
interrupt vector encoding, 21-6
interrupt vector generation, 21-6
masking interrupt sources, 21-5
MCC relative priority, 21-5
port C interrupts, 21-8
programming model, 21-9
registers, 21-9
SCC relative priority, 21-5
SCPRR_H, 21-9
SCPRR_L, 21-10, 38-11
SICR, 21-9
SIEXR, 21-15
SIMR_H, 21-13
SIPNR_H, 21-11
SIPNR_L, 21-12
SIVEC, 21-14

T

Target-disconnect, *see* PCI interface
Target-initiated termination
description, 15-4, 15-61
PCI status register, 15-62
TCN (timer counter registers), 25-8
TCR, 6-13
TCR (timer capture registers), 25-8
TDM,, 23-1
TER (timer event registers), 25-8
Termination
PCI
completion, 15-61
master-abort termination, 15-61
target-disconnect, 15-4, 15-61
target-initiated, 15-62
termination of PCI transaction, 15-60
timeout, 15-61
Test and configuration signals, *see* Signals, 19-8
TFCR (Tx buffer function code register)
overview, 27-15
TGCR (timer global configuration registers), 25-4
Three-speed Ethernet controller
AN advertisement register (ANA), 13-101
AN link partner ability next page register (ANLPANP), 13-106
AN link partner base page ability register (ANLPBPA), 13-103
AN next page transmit register (ANNPT), 13-105
application information, 13-135
attribute extract length and extract index register (ATTRELI), 13-97
attribute register (ATTR), 13-96
buffer descriptors, 13-129
carry mask register one (CAM1), 13-92
carry mask register two (CAM2), 13-93
carry register one (CAR1), 13-90
carry register two (CAR2), 13-91
connecting to physical interfaces, 13-110
control register (CR), 13-99
control/status registers, 2-63, 13-21

current receive buffer descriptor pointer (CRBPTR), 13-43
current transmit buffer descriptor pointer (CTBPTR), 13-36
data buffer descriptor, 13-129
destination address recognition, 13-123
detailed memory map—control/status registers, 13-15
DMA control register (DMACTRL), 13-29
drop frame counter (TDRP), 13-85
error disabled register (EDIS), 13-25
error-handling procedure, 13-128
ethernet control register (ECNTRL), 13-26
excessive collision packet counter (TXCL), 13-84
extended status register (EXST), 13-107
FIFO transmit starve register (FIFO_TX_STARVE), 13-33
FIFO transmit starve shutoff register (FIFO_TX_STARVE_SHUTOFF, 13-33
FIFO transmit threshold register (FIFO_TX_THR), 13-32
flow control, 13-126
frame recognition, 13-123
functional description, 13-110
gigabit Ethernet channel operation, 13-118
gigabit Ethernet frame reception, 13-122
gigabit ethernet frame transmission, 13-120
gigabit media-independent interface (GMII), 13-111
GMII interface mode, 13-139
group address registers 0–7 (GADDR*n*), 13-95
half-duplex register (HAFDUP), 13-52
hardware controlled initialization, 13-118
hash function registers, 13-95
hash table algorithm, 13-125
individual address registers 0–7 (IADDRn), 13-95
initialization sequence, 13-118
interface mode configuration, 13-135
interface status register (IFSTAT), 13-58
internal and external loop back, 13-128
inter-packet gap time, 13-127
inter-packet gap/inter-frame gap register (IPGIFG), 13-51
interrupt event register (IEVENT), 13-21
interrupt handling, 13-126
interrupt mask register (IMASK), 13-24
introduction, 13-1
jabber frame counter (TJBR), 13-86
jitter diagnostics register (JD), 13-108
MAC
  configuration 1 register (MACCFG1), 13-48
  configuration 2 register (MACCFG2), 13-50
  configuring, 13-46
  controlling packet flow, 13-47
  controlling PHY links, 13-48
  CSMA/CD controlling, 13-46
  functionality, 13-45
  handling packet collisions, 13-46
MAC registers, 13-48-13-61
maximum frame length register (MAXFRM), 13-54
maximum receive buffer length register (MRBL), 13-43
media-independent interface (MII), 13-111
memory map/register definition, 13-14
memory-mapped register descriptions, 13-21
MIB registers, 13-61-13-94
MII
  interface mode, 13-135
  management address register (MIIMADD), 13-56
  management command (MIIMCOM), 13-55
  management configuration register (MIIMCFG), 13-54
  management control register (MIIMCON), 13-57
  management indicator register (MIIMIND), 13-58
MII management status register (MIIMSTAT), 13-57
minimum frame length register (MINFLR), 13-28
modes of operation, 13-8
op-level module memory map, 13-15
overview, 1-17, 13-7
pause time value Register (PTV), 13-28
receive
  alignment error counter (RALN), 13-71
  broadcast packet counter (RBCA), 13-68
  byte counter (RBYT), 13-66
  carrier sense error counter (RCSE), 13-73
  code error counter (RCDE), 13-72
  control frame packet counter (RXCF), 13-69
  control register (RCTRL), 13-41
  descriptor base address register (RBASE), 13-45
  dropped packet counter (RDRP), 13-76
  fragments counter (RFRG), 13-75
  frame length error counter (RFLR), 13-72
  jabber counter (RJBR), 13-76
  multicast packet counter (RMCA), 13-68
  oversize packet counter (ROVR), 13-74
  packet counter (RPKT), 13-66
  pause frame packet counter (RXPF), 13-70
  status register (RSTAT), 13-41
  undersize packet counter (RUND), 13-74
  unknown OPcode packet counter (RXUO), 13-70
receive buffer descriptor (RxBD), 13-133
receive control and status registers, 13-40-13-45
reduced gigabit media-independent interface (RGMII), 13-112
reduced ten-bit interface (RTBI), 13-114
RMON support, 13-123
RxBD data length register (RBDLEN), 13-42

signals, 13-9-13-14
BIDI (input/output), 13-12
detailed descriptions, 13-11-13-15
EC_GTX_CLK125—Gigabit Transmit 125 MHz Source (input), 13-12
EC_MDC—Management Data Clock (output), 13-12
EC_MDIO—Management Data input/output, 13-12
TSEC_CRS—Carrier Sense (input), 13-11
TSEC*n*_COL—Collision (input), 13-11
TSEC*n*_GTX_CLK—Gigabit Transmit Clock (output), 13-11
TSEC*n*_RX_CLK—Receive Clock (input), 13-12
TSEC*n*_RX_DV—Receive Data Valid (input), 13-12
TSEC*n*_RX_ER—Receive Error (input), 13-13
TSEC*n*_RXD—Receive Data (input), 13-12
TSEC*n*_TX_CLK—Transmit Clock (input), 13-13
TSEC*n*_TX_EN—Transmit Data Valid (output), 13-14
TSEC*n*_TX_ER—Transmit Error (output), 13-14
TSEC*n*_TXD—Transmit Data (output), 13-13
soft reset and reconfiguring procedure, 13-119
station address part 1 register (MACSTNADDR1), 13-59
station address part 2 register (MACSTNADDR2), 13-60
status register (SR), 13-100
tansmit and receive
256- to 511-byte frame counter (TR511), 13-63
TBI control register (TBICON), 13-109
TBI MII set register descriptions, 13-98
TBI physical address register (TBIPA), 13-31
ten-bit interface (TBI), 13-98, 13-113
transmit
broadcast packet counter (TBCA), 13-79
byte counter (TBYT), 13-77
control frame counter (TXCF), 13-87
deferral packet counter (TDFR), 13-80
descriptor base address (TBASE), 13-37
excessive deferral packet counter (TEDF), 13-81
fragment counter (TFRG), 13-89
multicast packet counter (TMCA), 13-78
oversize frame counter (TOVR), 13-88
packet counter (TPKT), 13-78
pause control frame counter (TXPF), 13-80
single collision packet counter (TSCL), 13-82
status register (TSTAT), 13-35
total collision counter (TNCL), 13-84
undersize frame counter (TUND), 13-88
transmit and receive
1024- to 1518-byte frame counter (TRMAX), 13-64
128- to 255-byte fFrame counter (TR255), 13-62
1519- to 1522-byte VLAN frame counter (TRMGV), 13-65
512- to 1023-byte frame counter (TR1K), 13-64
64-Byte frame counter (TR64), 13-61
transmit and receive 65- to 127-byte frame counter (TR127), 13-62
Transmit Control Register (TCTRL), 13-34
transmit data buffer descriptor (TxBD), 13-131
transmit FCS error counter (TFCS), 13-86
TSEC FIFO control and status registers, 13-31
TSEC general control and status registers, 13-21
TSEC transmit control and status registers, 13-34
TxBD data length register (TBDLEN), 13-36
user initialization, 13-118
Timeout, PCI transaction, 15-61
Timers
block diagram, 25-1
bus monitoring, 25-3
cascaded mode block diagram, 25-4
features, 25-2
general-purpose units, 25-2
pulse measurement, 25-3
Time-slot assigner
connecting to the TSA, 22-7
Time-slot assigner (TSA)
synchronization in transparent mode, 31-6
Timing
SCC timing, controlling, 27-18
Timing diagrams
DRAM/EDO, self-refresh in sleep and suspend modes, 9-49
TLBs (translation lookaside buffers) coherency, 5-27
TM_CMD (RISC timer command) register, 20-38
TMR (timer mode registers), 25-6
TODR (transmit-on-demand register)
AppleTalk mode, 32-4
overview, 27-10
TOSEQ (transmit out-of-sequence) register, 28-10
Trace buffer
overview, 19-31
register descriptions, 19-18-19-25
Traced data (TBCR{TRSEL}), 19-31
Transactions
error transactions, 15-74
PCI bus
fast back-to-back transactions, 15-64
read transactions, 15-58
write transactions, 15-57
PCI transaction termination, 15-60
retry PCI transactions, 15-62
Transparent mode

achieving synchronization, 31-3
commands, 31-7
DSR receiver SYNC pattern lengths, 31-3
end of frame detection, 31-6
error handling, 31-8
fast communications controllers (FCCs)
features list, 41-1
receive operation, 41-2
synchronization
achieving, 41-2
example, 41-4
external signals, 41-3
in-line pattern, 41-3
transmit operation, 41-2
frame reception, 31-3
frame transmission, 31-2
inherent synchronization, 31-6
in-line synchronization, 31-6
overview, 31-1
programming example, 31-14
RxBD, 31-9
synchronization signals, 31-4
synchronization, user-controlled, 31-5
transmit synchronization, 31-3
TxBD, 31-11
$\overline{\text{TRDY}}$ (target ready) signal, 15-13, 15-53, 15-60
Trigger out source register (TOSR), 19-27
TRR (timer reference registers), 25-7
TSR, 6-14
TSTATE (internal transmitter state) register, 33-11
Turnaround cycle and PCI bus, 15-57

U

UART mode
commands, 28-6
control character insertion, 28-10
data handling, character and message-based, 28-5
error reporting, 28-6
features list, 28-2
fractional stop bits, 28-11
handling errors, 28-12
hunt mode, 28-10
normal asynchronous mode, 28-3
overview, 28-1
parameter RAM, 28-4
programming example, 28-23
RxBD, 28-15
S-records loader application, 28-24
status reporting, 28-6
synchronous mode, 28-3
TxBD, 28-19
User global control register 0 (UPMGC0), 6-51
User local control A registers (UPMLCa0–UPMLCa3), 6-52
User local control B registers (UPMLCb0–UPMLCb3), 6-53
USPRG0, 6-22

W

Watchpoint monitor
address mask register (WMAMR), 19-14
address register (WMAR), 19-13
control register 0 (WMCR0), 19-11
control register 1 (WMCR1), 19-12
general, 19-30
performance monitor events, 19-30
register descriptions, 19-11
status register (WMSR), 19-17
trace buffer
address mask register (TBAMR), 19-21
address register (TBAR), 19-20
control register 0 (TBCR0), 19-18
control register 1 (TBCR1), 19-20
status register (TBSR), 19-22
transaction mask register (TBTMR), 19-22
trace buffer access
control register (TBACR), 19-23
data high register (TBADHR), 19-24
data register (TBADR), 19-25
transaction mask register (WMTMR), 19-15

X

XER, 6-3, 6-8