MPC8560 PowerQUICC III™ Integrated Communications Processor Reference Manual

MPC8560RM
Rev. 1
07/2004

freescale™
semiconductor

How to Reach Us:

USA/Europe/Locations Not Listed:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405,
Denver, Colorado 80217
1-480-768-2130
(800) 521-6274

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

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

Home Page:
www.freescale.com

Learn More: For more information about Freescale Semiconductor products, please visit www.freescale.com

MPC8560RM
Rev. 1
07/2004

Contents

Contents

Chapter 2
Memory Map

Contents

Contents

Contents

Contents

Chapter 6
Core Register Summary

Contents

Contents

Chapter 7
L2 Look-Aside Cache/SRAM

Contents

Part III
Memory and I/O Interfaces

Chapter 8
e500 Coherency Module

Contents

Chapter 9
DDR Memory Controller

Contents

Chapter 10
Programmable Interrupt Controller

Contents

Contents

Chapter 11
I^2C Interface

Contents

Chapter 12
Local Bus Controller

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 14
DMA Controller

Contents

Contents

Contents

Contents

Contents

Chapter 16
RapidIO Interface

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 18
Performance Monitor

Contents

Contents

Contents

Contents

Chapter 23
CPM Multiplexing

Chapter 24
Baud-Rate Generators (BRGs)

Contents

Contents

Contents

Chapter 30
SCC BISYNC Mode

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 36
AAL1 Circuit Emulation Service

Contents

Contents

Contents

Chapter 39
Inverse Multiplexing for ATM (IMA)

Contents

Contents

Contents

Chapter 40
CPM Ethernet Controller

Contents

Contents

Chapter 44
CPM I^2C Controller

Contents

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Table Number	Title	Page Number

About This Book

The primary objective of this reference manual is to define the functionality of the MPC8560. 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 contains an embedded PowerPC™ core. 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. This book is intended as a companion to the *PowerPC e500 Core Complex Reference Manual*.

Audience

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

Organization

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

Part I, "Overview," describes the many features of the MPC8560 integrated communications processor at an overview level. The following chapters are included:

- Chapter 1, "Overview," provides a high-level description of features and functionality of the MPC8560 integrated communications processor. It describes the MPC8560, its interfaces, and its programming model. The functional operation of the MPC8560 with emphasis on peripheral functions is also described.
- 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 the sections detailing descriptions of each.
- Chapter 3, "Signal Descriptions," provides a listing of all the external signals, cross-references for signals that serve multiple functions, output signal states at reset, and reset configuration signals (and the modes they define).

- Chapter 4, “Reset, Clocking, and Initialization,” describes the hard and soft resets, the power-on reset sequence, power-on reset (POR) configuration, clocking, and initialization of the MPC8560.

Part II, “e500 Core Complex and L2 Cache,” describes the many features of the MPC8560 core processor at an overview level and the interaction between the core complex and the L2 cache. The following chapters are included:

- Chapter 5, “Core Complex Overview,” provides an overview of the e500 core processor and the L1 caches and MMU that, together with the core, comprise the core complex.
- Chapter 6, “Core 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.

Part III, “Memory and I/O Interfaces,” defines the memory and I/O interfaces of the MPC8560 and how these blocks interact with one another and with other blocks on the device. The following chapters are included:

- Chapter 8, “e500 Coherency Module,” defines the e500 coherency module and how it facilitates communication between the e500 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 override, crawl mode, and ECC error injection support rapid system debug.
- Chapter 10, “Programmable Interrupt Controller,” describes the embedded programmable interrupt controller (PIC) of the 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, “I^2C 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 Controllers," describes the 2 three-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/PCI-X Bus Interface," describes the PCI/PCI-X controller of the MPC8560.
- Chapter 16, "RapidIO Interface," describes the RapidIO controller of the MPC8560.

Part IV, "Global Functions and Debug," defines other global blocks of the MPC8560. The following chapters are included:

- 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. Note that the MPC8560 performance monitor is similar to but separate from the performance monitor implemented on the e500 core.
- Chapter 19, "Debug Features and Watchpoint Facility," describes the debug features and watchpoint monitor of the MPC8560.

Part V, “CPM Features,” defines the CPM blocks of the MPC8560. The following chapters are included:

- Chapter 20, “Communications Processor Module Overview,” provides a high-level summary of the MPC8560 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 two MPC8560 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 also 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, “Inverse Multiplexing for ATM (IMA),” describes the MPC8560’s implementation of inverse multiplexing for ATM.
- Chapter 40, “CPM Ethernet Controller,” describes the fast Ethernet controller in the CPM.
- Chapter 41, “FCC HDLC Controller,” describes the FCC HDLC controller of the MPC8560.
- Chapter 42, “FCC Transparent Controller,” describes how the FCC transparent controller functions as a high-speed serial-to-parallel and parallel-to-serial converter.
- Chapter 43, “Serial Peripheral Interface (SPI),” describes the serial peripheral interface (SPI) of the MPC8560 CPM.
- Chapter 44, “CPM I2C Controller,” describes the I^2C controller of the CPM.
- Chapter 45, “Parallel I/O Ports,” describes the four general-purpose I/O ports of the CPM.
- Appendix A, “Revision History,” lists the major differences between revisions of the *MPC8560 PowerQUICC III Integrated Communications Processor Reference Manual*.
- This document also contains the following indexes:
  - Register index. Contains listings of all memory-mapped registers implemented by the integrated logic. It does not include the following:
    - e500 core registers. These registers are listed in the general index under *e500 core, registers*.
    - CPM registers. These registers are listed in the CPM index.

  - Configuration header registers defined by the PCI specification. These registers are listed under *PCI/PCI-X controller, register descriptions.*
- General index. Contains all entries except those for the CPM and the memory-mapped registers.
- CPM index. Includes all index references to CPM functionality.

- This reference manual also includes a glossary.

Suggested Reading

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

General Information

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

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

Related Documentation

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

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

- Hardware specifications—Hardware specifications provide specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations.
- Technical summaries—Each device has a technical summary that provides an overview of its features. This document is roughly equivalent to the overview (Chapter 1) of an implementation's reference or user's manual.
- Application notes—These short documents address specific design issues useful to programmers and engineers working with Freescale Semiconductor processors.

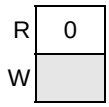
Additional literature is published as new processors become available. For a current list of documentation, refer to http://www.freescale.com.

Conventions

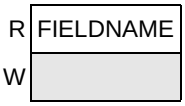
This document uses the following notational conventions:

cleared/set	When a bit takes the value zero, it is said to be cleared; when it takes a value of one, it is said to be set.
mnemonics	Instruction mnemonics are shown in lowercase bold.
italics	Italics indicate variable command parameters, for example, **bcctr***x*.
	Book titles in text are set in italics
	Internal signals are set in lowercase italics, for example, $\overline{core\ int}$
0x0	Prefix to denote hexadecimal number
0b0	Prefix to denote binary number
rA, **r**B	Instruction syntax used to identify a source GPR
rD	Instruction syntax used to identify a destination GPR
REG[FIELD]	Abbreviations for registers are shown in uppercase text. Specific bits, fields, or ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In some contexts, such as signal encodings, an unitalicized x indicates a don't care.
x	An italicized *x* indicates an alphanumeric variable.
n	An italicized *n* indicates a numeric variable.
¬	NOT logical operator
&	AND logical operator
|	OR logical operator
||	Concatenation, for example TCR[WPEXT]||TCR[WP]

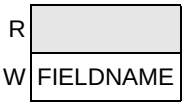
Indicates a reserved bit field in an e500 register. Although these bits can be written to as ones or zeros, they are always read as zeros.



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



Indicates a read-only bit field in a memory-mapped register.



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

Signal Conventions

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

Acronyms and Abbreviations

Table i contains acronyms and abbreviations used in this document.

Table 1. Acronyms and Abbreviated Terms

Term	Meaning
ADB	Allowable disconnect boundary
ATM	Asynchronous transfer mode
ATMU	Address translation and mapping unit
BD	Buffer descriptor
BIST	Built-in self test
BRI	Basic rate interface
BTB	Branch target buffer
BUID	Bus unit ID
CAM	Content-addressable memory
CCB	Core complex bus
CCSR	Configuration control and status register
CEPT	Conference des administrations Europeanes des Postes et Telecommunications (European Conference of Postal and Telecommunications Administrations).

Table 1. Acronyms and Abbreviated Terms (continued)

Term	Meaning
COL	Collision
CPM	Communication processor module
CRC	Cyclic redundancy check
CRS	Carrier sense
DDR	Double data rate
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory
DUART	Dual universal asynchronous receiver/transmitter
EA	Effective address
ECC	Error checking and correction
ECM	e500 coherency module
EEST	Enhanced Ethernet serial transceiver
EHPI	Enhanced host port interface
EPROM	Erasable programmable read-only memory
FCS	Frame-check sequence
GCI	General circuit interface
GMII	Gigabit media independent interface
GPCM	General-purpose chip-select machine
GPIO	General-purpose I/O
GPR	General-purpose register
GUI	Graphical user interface
HDLC	High-level data link control
I^2C	Inter-integrated circuit
IDL	Inter-chip digital link
IEEE	Institute of Electrical and Electronics Engineers
IPG	Interpacket gap
IrDA	Infrared Data Association
ISDN	Integrated services digital network
ITLB	Instruction translation lookaside buffer
IU	Integer unit
JTAG	Joint Test Action Group

Table 1. Acronyms and Abbreviated Terms (continued)

Term	Meaning
LAE	Local access error
LAW	Local access window
LBC	Local bus controller
LIFO	Last-in-first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
LSU	Load/store unit
MAC	Multiply accumulate, media access control
MDI	Medium-dependent interface
MESI	Modified/exclusive/shared/invalid—cache coherency protocol
MII	Media independent interface
MMU	Memory management unit
MSB	Most-significant byte
msb	Most-significant bit
NMSI	Nonmultiplexed serial interface
No-op	No operation
OCeaN	On-chip network
OSI	Open systems interconnection
PCI	Peripheral component interconnect
PCI/X	Abbreviation used to describe operation for both the PCI and PCI-X bus functionality
PCI-X	PCI extended
PCMCIA	Personal Computer Memory Card International Association
PCS	Physical coding sublayer
PIC	Programmable interrupt controller
PMA	Physical medium attachment
PMD	Physical medium dependent
POR	Power-on reset
PRI	Primary rate interface
RGMII	Reduced gigabit media independent interface
RISC	Reduced instruction set computing
RIO	Abbreviation occasionally used to refer to the RapidIO interface

Table 1. Acronyms and Abbreviated Terms (continued)

Term	Meaning
RTOS	Real-time operating system
RWITM	Read with intent to modify
RWM	Read modify write
Rx	Receive
RxBD	Receive buffer descriptor
SCC	Serial communication controller
SCP	Serial control port
SDLC	Synchronous data link control
SDMA	Serial DMA
SFD	Start frame delimiter
SI	Serial interface
SIU	System interface unit
SMC	Serial management controller
SNA	Systems network architecture
SPI	Serial peripheral interface
SPR	Special-purpose register
SRAM	Static random access memory
TAP	Test access port
TBI	Ten-bit interface
TDM	Time-division multiplexed
TLB	Translation lookaside buffer
TSA	Time-slot assigner
TSEC	Three-speed Ethernet controller
Tx	Transmit
TxBD	Transmit buffer descriptor
UART	Universal asynchronous receiver/transmitter
UPM	User-programmable machine
USB	Universal serial bus
UTP	Unshielded twisted pair
VA	Virtual address
ZBT	Zero bus turnaround

Part I
Overview

Part I describes the many features of the MPC8560 integrated communications processor at an overview level. The following chapters are included:

- 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 processor. It describes the MPC8560, its interfaces, and its programming model. The functional operation of the MPC8560 with emphasis on peripheral functions is also described.

Chapter 2, “Memory Map,” describes the MPC8560 memory map. An overview of the local address map is followed by a description of how local access windows are used to define the local address map. The inbound and outbound address translation mechanisms used to map to and from external memory spaces are described next. Finally, the configuration, control, and status registers are described, including a complete listing of all memory mapped registers with cross references to the sections detailing descriptions of each.

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

Chapter 4, “Reset, Clocking, and Initialization,” describes the hard and soft resets, 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 integrates the processing power for networking and communications peripherals, resulting in higher device performance. The MPC8560 contains an embedded PowerPC™ core. The MPC8560 is a member of a 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 the features and functionality of the MPC8560 integrated microprocessor.

1.1 Introduction

Freescale Semiconductor's leading PowerQUICC III architecture integrates two processing blocks—a high-performance embedded e500 core and the communications processor module (CPM). The e500 core implements the enhanced Book E instruction set architecture and provides unprecedented levels of hardware and software debugging support.

The CPM of the MPC8560 supports 3 fast serial communications channels (FCCs) for 155-Mbps ATM and fast Ethernet and up to 256 full-duplex, time-division-multiplexed (TDM) channels using 2 multi-channel controllers (MCCs). In addition, the CPM supports four serial communications controllers (SCCs), one serial peripheral interface (SPI), and one I^2C interface.

In addition, the MPC8560 offers 256 Kbytes of L2 cache, 2 integrated 10/100/1Gb three-speed Ethernet controllers (TSECs), a DDR SDRAM memory controller, a 64-bit PCI/PCI-X controller, an 8-bit RapidIO port, a programmable interrupt controller (PIC), an I^2C controller, a 4-channel DMA controller, and a general-purpose I/O port. The high level of integration in the MPC8560 simplifies 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 features of the MPC8560.

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

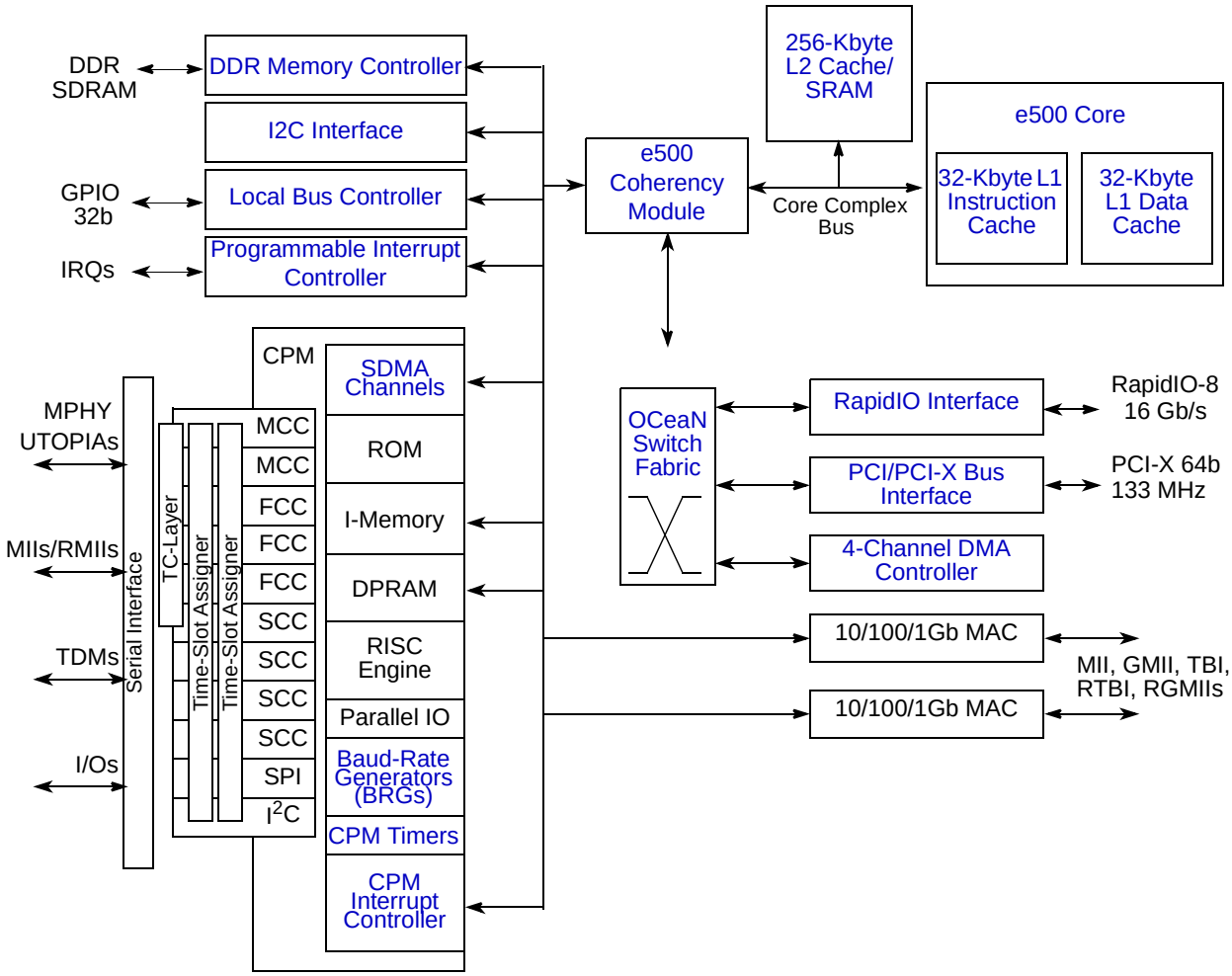


Figure 1-1. MPC8560 Block Diagram

1.2.1 Key Features

The following is an overview of the MPC8560 feature set:

- High-performance, 32-bit Book E–enhanced core that implements the PowerPC architecture
  - 32-Kbyte L1 instruction cache and 32-Kbyte L1 data cache with parity protection. Caches can be locked entirely or on a per-line basis, with separate locking for instructions and data.
  - Signal-processing engine (SPE) auxiliary processing unit (APU) provides an extensive instruction set for vector (64-bit) integer, single-precision floating-point, and fractional operations. These instructions use both the upper and lower words of the 64-bit GPRs as they are defined by the SPE APU.

— The single-precision floating-point (SPFP) APU provides an instruction set for single-precision (32-bit) floating-point instructions.
— Memory management unit (MMU) especially designed for embedded applications
— Enhanced hardware and software debug support
— Performance monitor facility (similar to but different from the MPC8560 performance monitor described in Chapter 18, “Performance Monitor.”)

The e500 defines features that are not implemented on the MPC8560. It also generally defines some features that the MPC8560 implements more specifically. An understanding of these differences can be critical to ensure proper operations. These differences are summarized in Section 5.14, “MPC8560 Implementation Details,” in the *MPC8560 PowerQUICC III Integrated Communications Processor Reference Manual*.

Section 1.3.1, “e500 Core Overview,” includes a comprehensive list of e500 core features.

- 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) that 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 8 TDM interfaces
  — Four full-duplex serial communications controllers (SCCs) that 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 (TSA) 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
  - Freescale Semiconductor 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-Kbyte L2 cache/SRAM
  - Can be configured as follows:
    - Full cache mode (256-Kbyte cache)
    - Full memory-mapped SRAM mode (256-Kbyte SRAM mapped as a single 256-Kbyte block or two 128-Kbyte blocks)
    - Half SRAM and half cache mode (128-Kbyte cache and 128-Kbyte memory-mapped SRAM)
  - Full error checking and correction (ECC) support on 64-bit boundary in both cache and SRAM modes
  - 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 (stashing).
  - Eight-way set-associative cache organization (1024 sets of 32-byte cache lines)

— Supports locking the entire cache or selected lines. Individual line locks are 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
— SRAM features include the following:
  – I/O devices access SRAM regions by marking transactions as snoopable (global).
  – Regions can reside at any aligned location in the memory map.
  – Byte-accessible ECC is protected using read-modify-write transactions accesses for smaller than cache-line accesses

- Address translation and mapping unit (ATMU)
  — Eight local access windows define mapping within local 32-bit address space
  — Inbound and outbound ATMUs 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 333-MHz data rate
  — Four banks of memory supported, each up to 1 Gbyte
  — DRAM chip configurations from 64 Mbits 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 signal
  — Registered DIMM support
  — Fast memory access through JTAG port
  — 2.5-V 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 signaling
- 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)
  - Programming model is compliant with the OpenPIC architecture.
  - 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 programmable interrupt controller
  - Four global high resolution timers/counters that can generate interrupts
  - Supports 22 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 through 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 at up to 166 MHz
  - Eight chip selects support eight external slaves
  - Four- and 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 (UPMs)
    - Dedicated single data rate SDRAM controller
  - Parity support
  - Default boot ROM chip select with configurable bus width (8-,16-, or 32-bit)
- Two three-speed (10/100/1Gb) Ethernet controllers (TSECs)
  - Dual IEEE 802.3, 802.3u, 802.3x, 802.3z, 802.3ac, 802.3ab compliant controllers
  - Support for different Ethernet physical interfaces:
    - 10/100/1Gb IEEE 802.3 GMII
    - 10/100 Mbps IEEE 802.3 MII
    - 10-Mbps IEEE 802.3 MII
    - 1-Gbps IEEE 802.3z TBI
    - 10/100/1Gb RGMII/RTBI
  - Full- and half-duplex support
  - Buffer descriptors are backward 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
  - Ability to force allocation of header information and buffer descriptors into L2 cache
- 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 of up to 256 bytes
- Integrated DMA controller
  - Four-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
  - Misaligned 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 controller
  - PCI 2.2 and PCI-X 1.0 compatible
  - 64- or 32-bit PCI port supports at 16 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
  - Supports posting of processor-to-PCI and PCI-to-memory writes
  - PCI 3.3-V compatible
  - Selectable hardware-enforced coherency

- Power management
  - Fully static 1.2-V CMOS design with 3.3- and 2.5-V I/O
  - Supports power saving modes: doze, nap, and sleep
  - Employs dynamic power management, which automatically minimizes power consumption of blocks when they are idle
- System performance monitor
  - Supports 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 8 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 sections 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 documents:

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

NOTE

The e500 defines features that are not implemented on the MPC8560. It also generally defines some features that the MPC8560 implements more specifically. An understanding of these differences can be critical to ensure proper operation. These differences are summarized in Section 5.14, “MPC8560 Implementation Details,” in the *MPC8560 PowerQUICC III Integrated Communications Processor Reference Manual*.

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

- Implements full Book E 32-bit architecture
- Implements additional instructions, registers, and interrupts defined by APUs. The SPE provides an extensive instruction set for 64-bit vector integer, single-precision floating-point, and fractional operations. The SPFP APU provides scalar (32-bit) single-precision, floating-point instructions.

NOTE

The SPE APU and SPFP APU functionality will be implemented in the MPC8540, the MPC8560 and in their derivatives (that is, in all PowerQUICC III devices). However, these instructions will not be supported in devices subsequent to PowerQUICC III. Freescale Semiconductor strongly recommends that use of these instructions be confined to libraries and device drivers. Customer software that uses SPE or SPFP APU instructions at the assembly level or that uses SPE intrinsics will require rewriting for upward compatibility with next-generation PowerQUICC devices.

Freescale Semiconductor offers a libmoto_e500 library that uses SPE and SPFP APU instructions. Freescale will also provide future libraries to support next generation PowerQUICC devices.

- L1 cache structure
  - 32-Kbyte, 32-byte line, eight-way set-associative instruction cache
  - 32-Kbyte, 32-byte line, eight-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
- Dual-dispatch superscalar
- Precise exception handling
- Seven-stage pipeline control

- Instruction unit
  - Twelve-entry instruction queue
  - Full hardware detection of interlocks
  - Dispatch up to two instructions per cycle
  - Dispatch serialization control
  - Register dependency resolution and renaming
- Branch unit (BU)
  - Dynamic branch prediction
  - Two-entry branch instruction queue (BIQ)
  - 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 execution units that perform the following:
  - Single-cycle add and subtract
  - Single-cycle shift and rotate
  - Single-cycle logical operations
  - Supports integer signal processing operations
- Multiple-cycle execution unit (MU)
  - Four-cycle latency for integer and floating-point multiplication (including integer, fractional, and both vector and scalar floating-point multiply instructions).
  - Variable-latency divide: 4, 11, 19, and 35 cycles for all Book E, SPE, and SPFP divide instructions. Note that the MU allows divide instructions to bypass the second two MU pipeline stages, freeing those stages for other MU instructions to execute in parallel.
  - Four-cycle floating-point multiply
  - Four-cycle floating-point add and subtract
- Signal processing engine APU (SPE APU). The SIMD capability provided by the 64-bit 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

  - Four-cycle integer pipelined multiplies
  - 4-, 11-, 19-, and 35-cycle integer divides
  - Four-cycle single instruction multiple data (SIMD) pipelined multiply-accumulate (MAC)
  - 64-bit accumulator for MAC operations
  - Single-precision floating-point operations
- Load/store unit (LSU)
  - Three-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
  - 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, one 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
    - Floating-point data exception
    - Floating-point round exception
    - Performance monitor
- Memory management unit (MMU)
  - Data L1 MMU
    - Four-entry, fully-associative TLB array for variable-sized pages
    - 64-entry, four-way set-associative TLB for 4-Kbyte pages
  - Instruction L1 MMU
    - Four-entry, fully-associative TLB array for variable-sized pages
    - 64-entry, four-way set-associative TLB for 4-Kbyte pages
  - Unified L2 MMU
    - 16-entry, fully-associative TLB array for variable-sized pages
    - 256-entry, two-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, 1.2-V design
  - Dynamic power management on the core minimizes power consumption of functional units, such as execution units, caches, and MMUs, when they are idle.
  - Core power-saving modes: core-halted and core-stopped
  - NAP, DOZE, and SLEEP bits in HID0 that can be used to assert *nap*, *doze*, and *sleep* core output signals to initiate power-saving modes at the integrated-device level
  - Internal clock multipliers of 2x, 2.5x, and 3x from bus clock
- 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 the networking and telecommunication markets. The MPC8560 CPM is a superset of the MPC8260 PowerQUICC II, with enhanced communications processor (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 in the MPC8560 CPM.

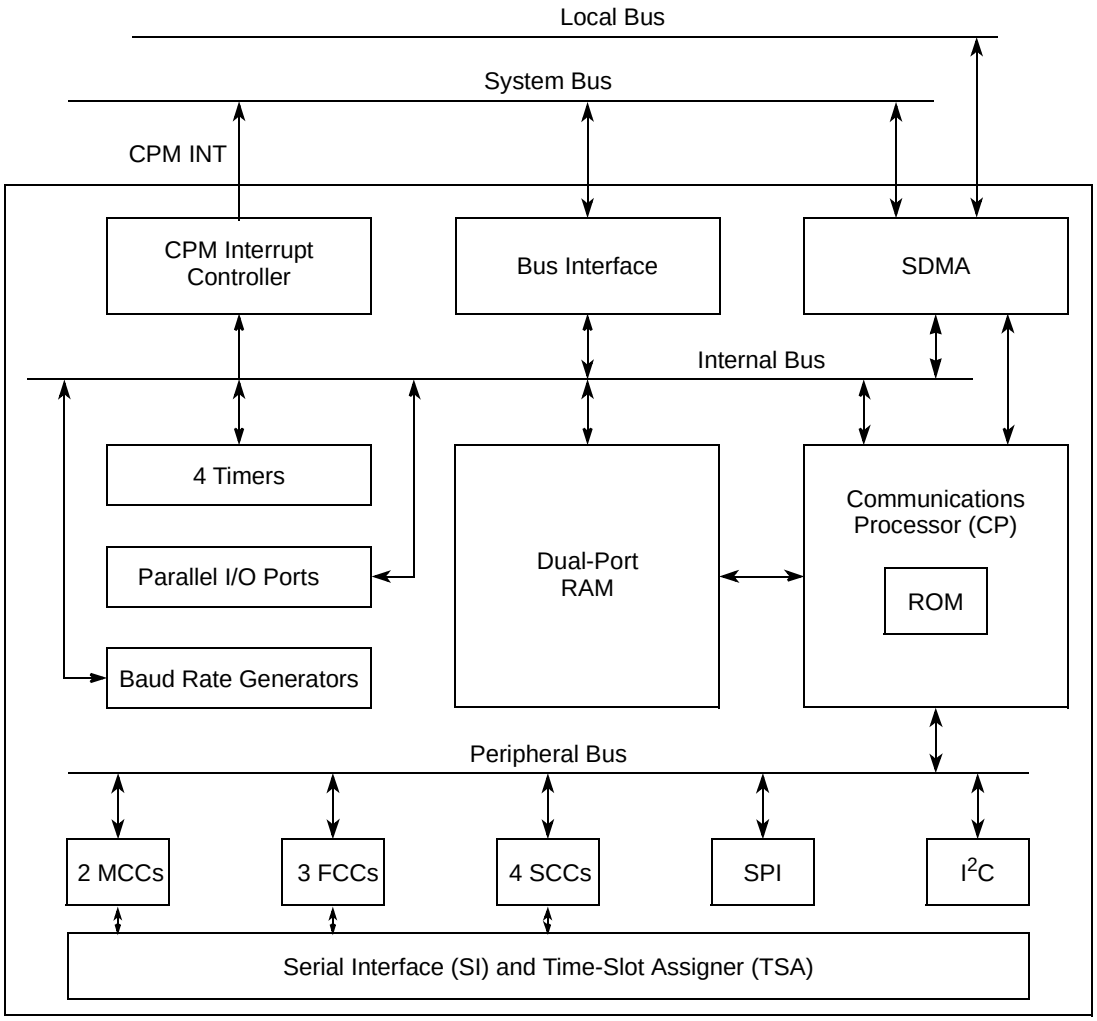


Figure 1-2. MPC8560 Communications Processor Module (CPM) Block Diagram

The following list summarizes the major features of the CPM:

- The 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 e500 coherency module (ECM), handling transfers simultaneously
- Three full-duplex, serial FCCs supporting ATM (155 Mbps) protocol through two UTOPIA L2 interfaces, 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 data and receive HDLC data, or vice-versa.
- Two MCCs capable of handling an aggregate of 256 64-Kbps HDLC or transparent channels, multiplexed on up to 8 TDM interfaces. The MCC also supports super-channels of rates higher than 64 Kbps and subchanneling of the 64-Kbps channels.
- Four full-duplex SCCs supporting high-level synchronous data link control, HDLC, local talk, UART, synchronous UART, BISYNC, and transparent
- SPI and I^2C bus controllers
- 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 that receive ATM traffic over standard serial protocols (T1, E1, xDSL) through their 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 also be divided into two 128-Kbyte arrays, one of which may be used as cache and the other as SRAM.

The memory controller for this array connects to the core complex bus (CCB) and communicates through 128-bit read and write buses to the e500 core and the MPC8560 system logic.

The on-chip memory unit contains:

- 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 8-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 (1 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

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 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. The replacement policy in each eight-way set is governed by a pseudo-LRU algorithm. The data is protected with ECC, and 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 or stores can occur to that line on the very next cycle.

The L2 status array maintains status bits for each line to determine the status of the line. Different combinations of these bits result in different L2 states. Note that because the cache is always write-through, there is no modified state. The status bits include the following:

- V—Valid
- 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 (stashing). 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 L2 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 cache 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 device.

1.3.5 DDR SDRAM Controller

The MPC8560 supports DDR-I SDRAM that operates at 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, and 512 Mbits, and 1 Gbit. 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]) 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 16 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 3 to 4 clock cycles from subsequent burst accesses that hit in an active page.

The MPC8560 supports 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 asserting the MCKE SDRAM signal on-the-fly to put the memory into a low-power sleep mode.

1.3.6 Programmable Interrupt Controller (PIC)

The programmable interrupt controller (PIC) 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 processor core and provides for 12 external interrupts (with fully nested interrupt delivery), 4 message interrupts, internal-logic driven interrupts, and 4 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 (IPI) communication is supported through the external interrupt and core reset signals of different processor cores on the same device. The four IPIs are only used for self-interrupt in a single-core device such as the MPC8560.

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 and efficient method of data exchange between devices. The synchronous, multiple-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, multiple-master 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 multiple-master bus; it 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 a 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 at the negation of the MPC8560 hardware reset signal. If enabled, the boot sequencer holds the MPC8560 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 (LBC)

The MPC8560 local bus controller (LBC) 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 LBC provides a synchronous DRAM (SDRAM) machine that supplies 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 (10/100/1Gb)

The MPC8560 has two on-chip three-speed Ethernet controllers (TSECs). The TSECs incorporate a media access control sublayer (MAC) that supports 10- and 100-Mbps, and 1-Gbps Ethernet/802.3 networks with MII, GMII, RGMII, RTBI, and TBI physical interfaces. The TSECs include 2-Kbyte receive and transmit FIFOs, and DMA functions.

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

The MPC8560 TSECs support programmable CRC generation and checking, RMON statistics, and jumbo frames of up to 9.6 Kbytes. Frame headers and buffer descriptors can be forced into the L2 cache to speed classification or other frame processing.

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 four-channel DMA controller allows chaining (both extended and direct) through local memory-mapped chain descriptors. Scattering, gathering, and misaligned transfers are supported. In addition, advanced capabilities such as stride transfers and complex transaction chaining are supported.

DMA transfers can be initiated by a single write to a configuration register. There is also support for external control of transfers using $\overline{\text{DMA\_DREQ}}$, $\overline{\text{DMA\_DACK}}$, 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 directly to the RapidIO port. Local attributes such as snoop and L2-write stashing can be specified by descriptors.

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

1.3.12 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 internally combines transactions up to 256 bytes.

An internal arbiter can be used to support up to five 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 *RapidIO Interconnect Specification, Revision 1.1*. 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. Because the interface is defined as a source-synchronous, double-data-rate, LVDS-signaling interconnect, the theoretical unidirectional peak bandwidth is 8 Gbps. Receive and transmit ports operate independently, resulting in an aggregate theoretical bandwidth of 16 Gbps.

1.3.13.1 RapidIO Message Unit

The 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 four power consumption modes: full-on, doze, nap, and sleep. The three low-power modes: doze, nap, and sleep, can be entered under software control in the e500 core or by external masters accessing a configuration register.

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

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

Sleep mode shuts down not only the e500 core, but all of the MPC8560 I/O interfaces as well. Only the interrupt controller and power management logic remain enabled so that the device can be awakened.

1.3.15 Clocking

The MPC8560 takes in the PCI_CLK/SYSCLK signal as an input to the device PLL and multiplies it by an integer from 1 to 16 to generate the core complex bus clock (the platform clock), which operates at the same frequency as the DDR DRAM data rate (for example, 266 or 333 MHz). The L2 cache also operates at this frequency. The e500 core uses the CCB clock as an input to its PLL, which multiplies it again by 2, 2.5, 3, or 3.5 to generate the core clock.

DLLs are used in the DDR SDRAM controller and the local bus memory controller (LBC) to generate memory clocks. Six differential clock pairs are generated for DDR SDRAMs. Two clock outputs are generated for the LBC.

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 device driving a clock with a frequency of 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 units (ATMUs). Both inbound and outbound translation windows are provided. The ATMUs 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. With the first method, data is processed by the MPC8560 CPM (as it is received and transmitted through the UTOPIAs, MIIs, and TDMs associated with the CPM) and the local bus. In the second method, data is received by any of the available I/O ports, sent through the on-chip switch fabric, and transmitted on the target I/O port without the use of the ECM. The third method by which data can be routed from any I/O port to any other I/O port is through the ECM.

1.4.1 Processing Between the CPM and Local Bus

In this case, the MPC8560 stores data in buffers that 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. Rx data is decoded when it arrives on the I/O port; the PDU is delineated from the incoming data stream.
2. Data is converted from its serial form into a parallel form and 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 as 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 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.

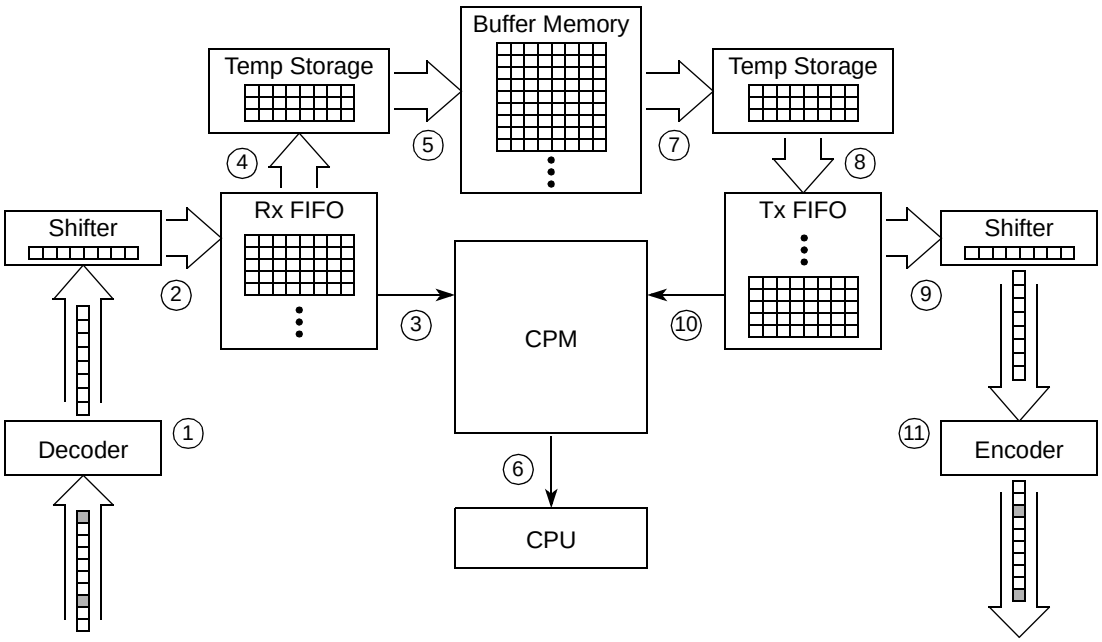


Figure 1-3. Data Processing in the CPM

1.4.2 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 ATMUs process 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.

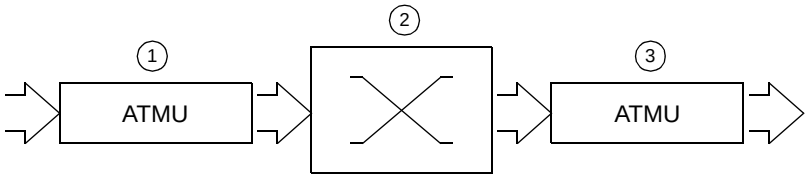


Figure 1-4. Processing Transactions Across the On-Chip Fabric

1.4.3 Data Processing with the e500 Coherency Module

Processing through 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 using the ECM 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 designated 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 other transactions (between the CPM and the local bus and across the on-chip fabric) are non-coherent.

Although 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 setting the appropriate bits in the FCC and SCC functional code registers, respectively.

1.5 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.

1.5.1 Device Configurations

The following are the three main system configurations for the MPC8560:

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

1.5.1.1 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 local memory without interfering with the operation of the e500 core.

Figure 1-5 shows a basic system configuration.

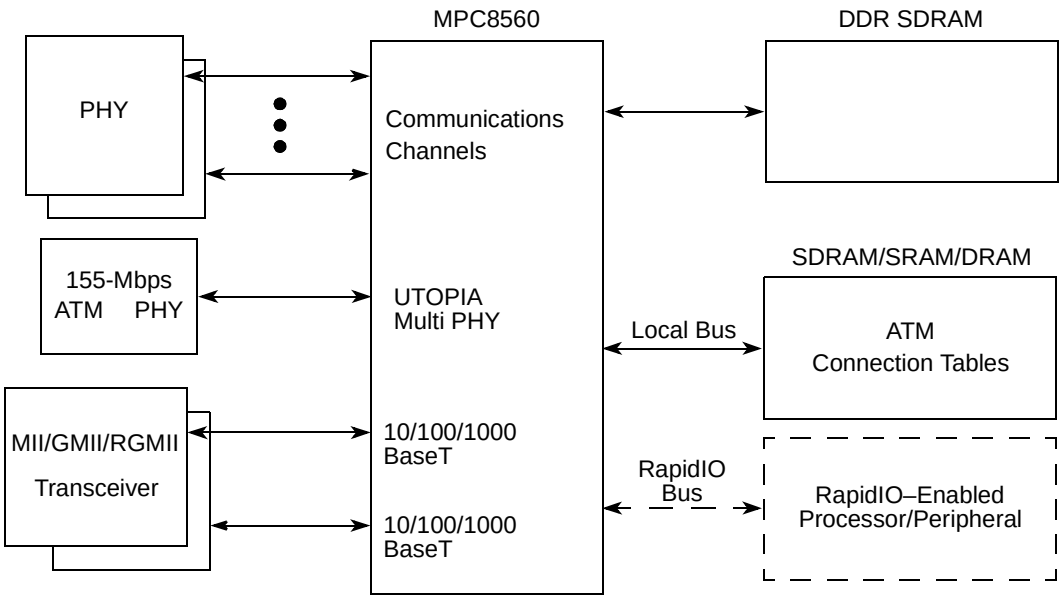


Figure 1-5. Basic System Configuration

1.5.1.2 Multiprocessor System

Figure 1-6 shows a multiprocessor system configuration.

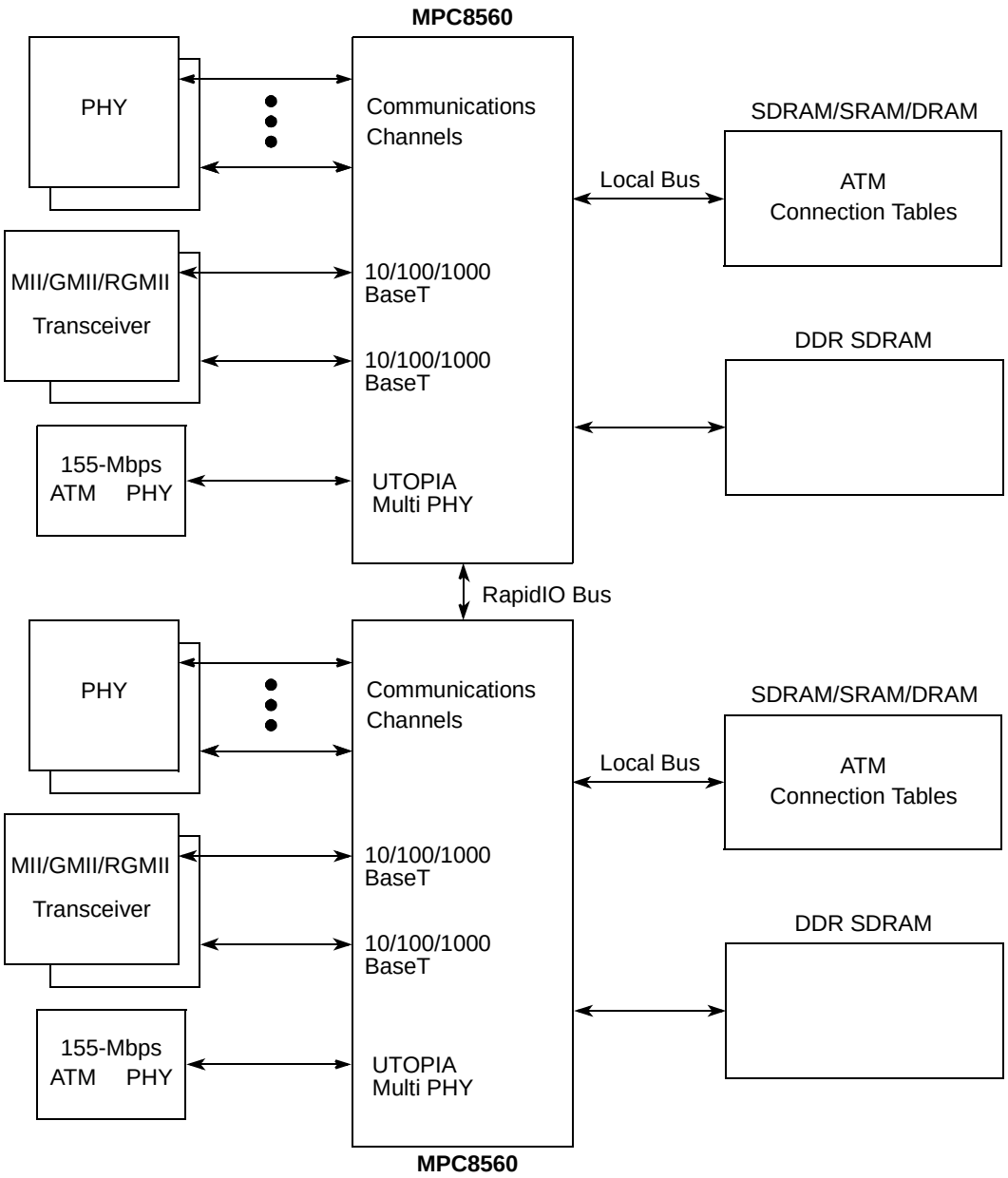


Figure 1-6. High-Performance Communications

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.

1.5.1.3 High-Performance System

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

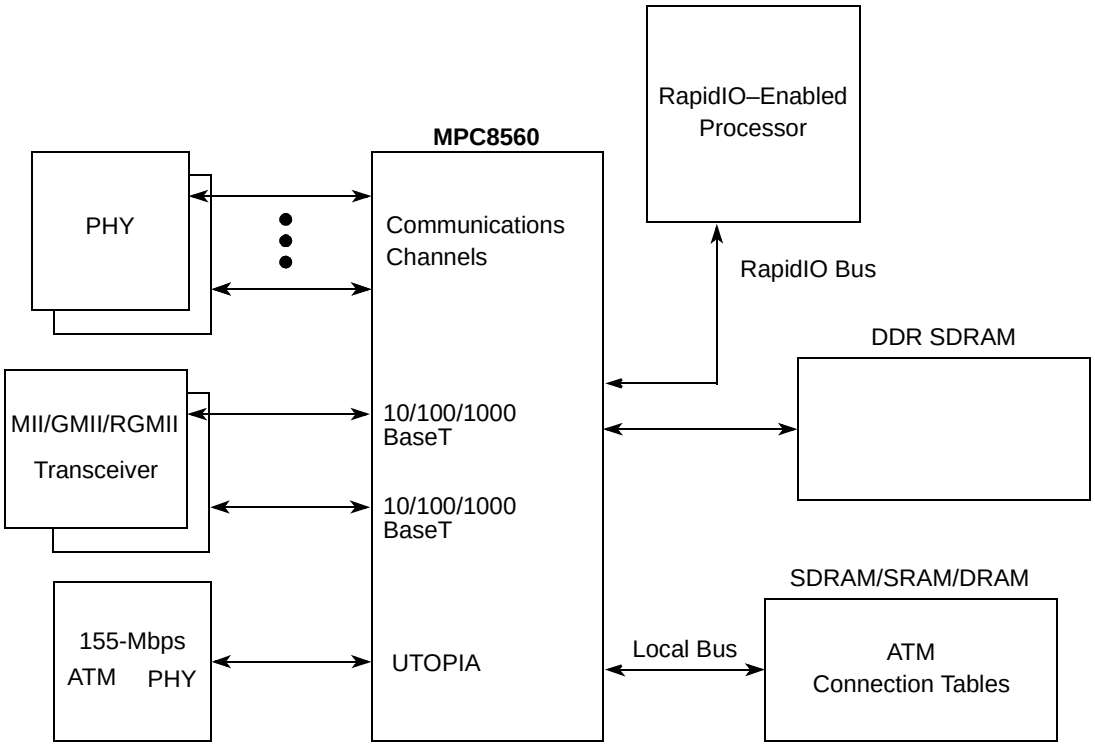


Figure 1-7. High-Performance System Microprocessor Configuration

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

1.5.2 Examples of Communications Systems

The 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

1.5.2.1 Remote Access Server

Figure 1-8 shows a remote access server configuration.

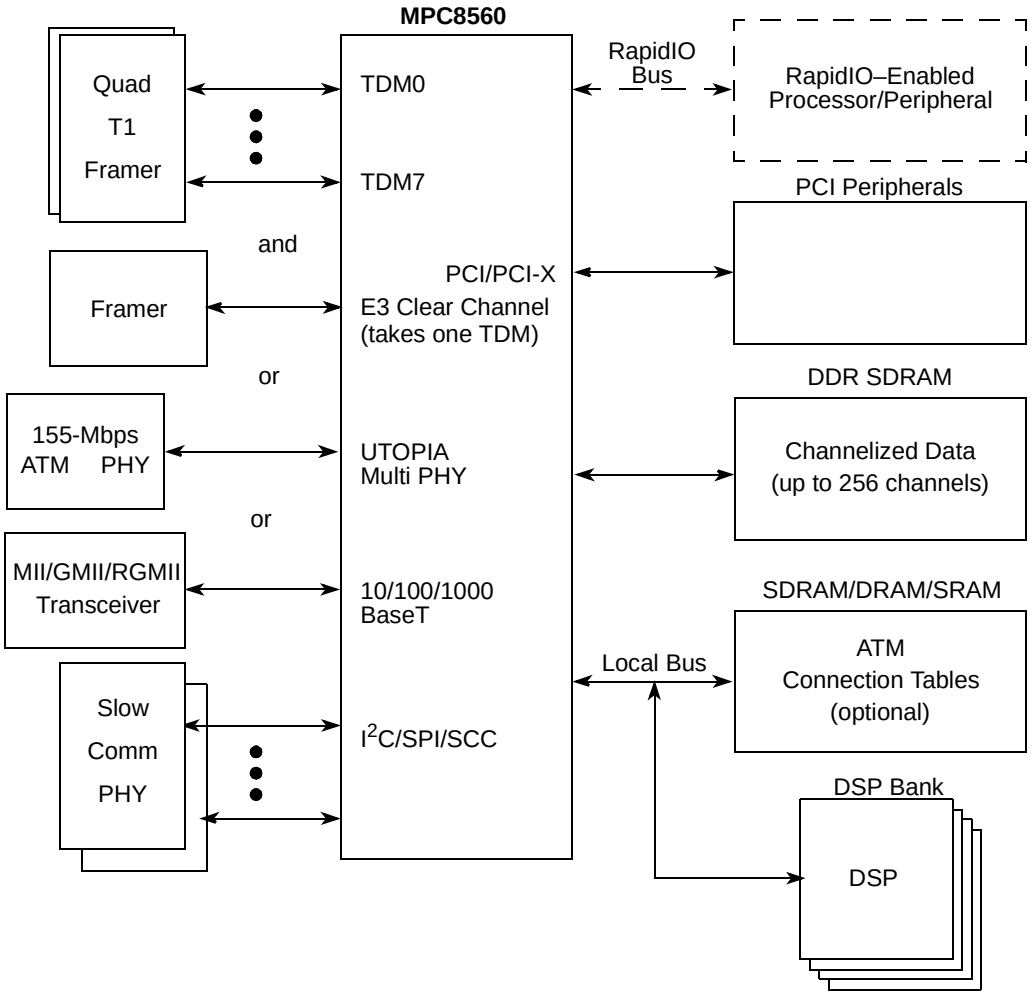


Figure 1-8. Remote Access Server Configuration

In this application, eight TDM ports are connected to external framers. In the MPC8560, each group of four ports supportS 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.

1.5.2.2 Regional Office Router

Figure 1-9 shows a regional office router configuration.

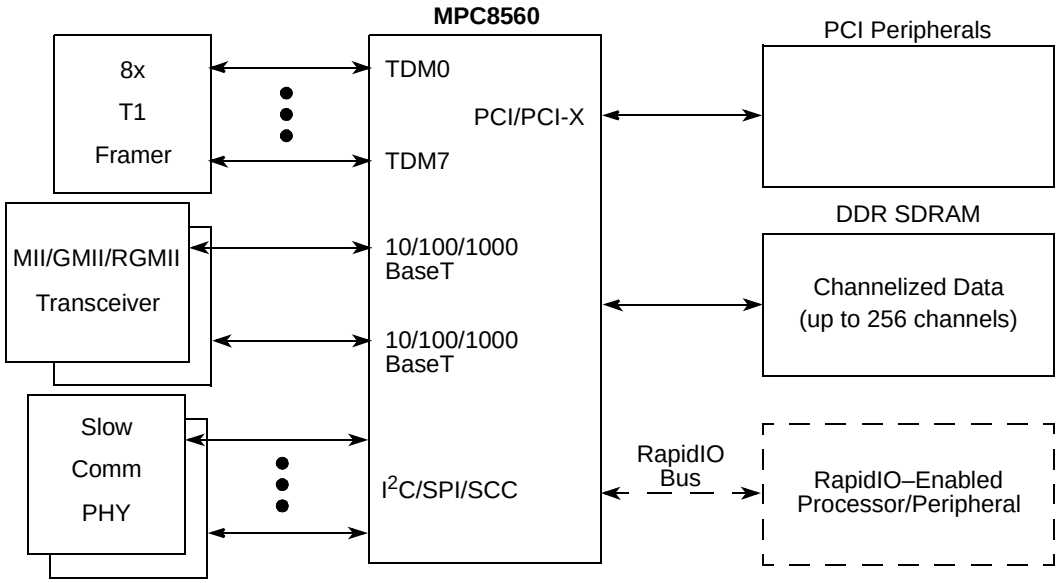


Figure 1-9. Regional Office Router Configuration

In this application, the MPC8560 is connected to up to 8 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. This application has two MII/GMII/RGMII ports for 10/100/1000BaseT LAN connections.

The SCC ports can be used for management in all of the examples.

1.5.2.3 LAN-to-WAN Bridge Router

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

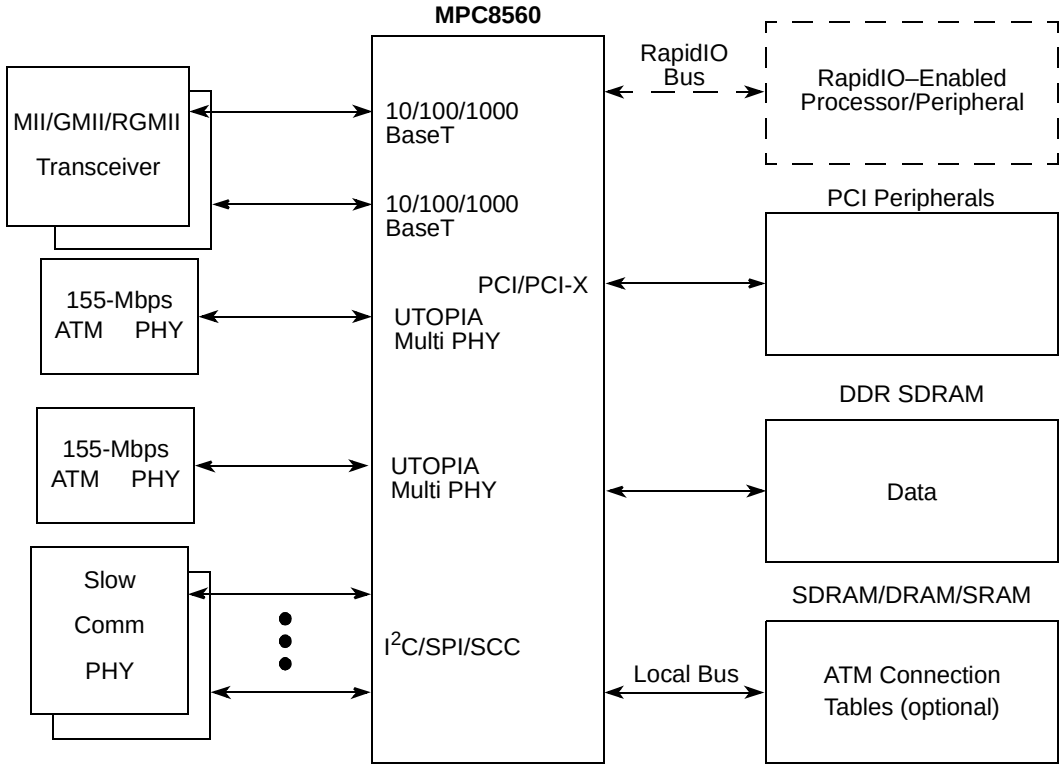


Figure 1-10. LAN-to-WAN Bridge Router Configuration

1.5.2.4 Cellular Base Station

Figure 1-11 shows a cellular base station configuration.

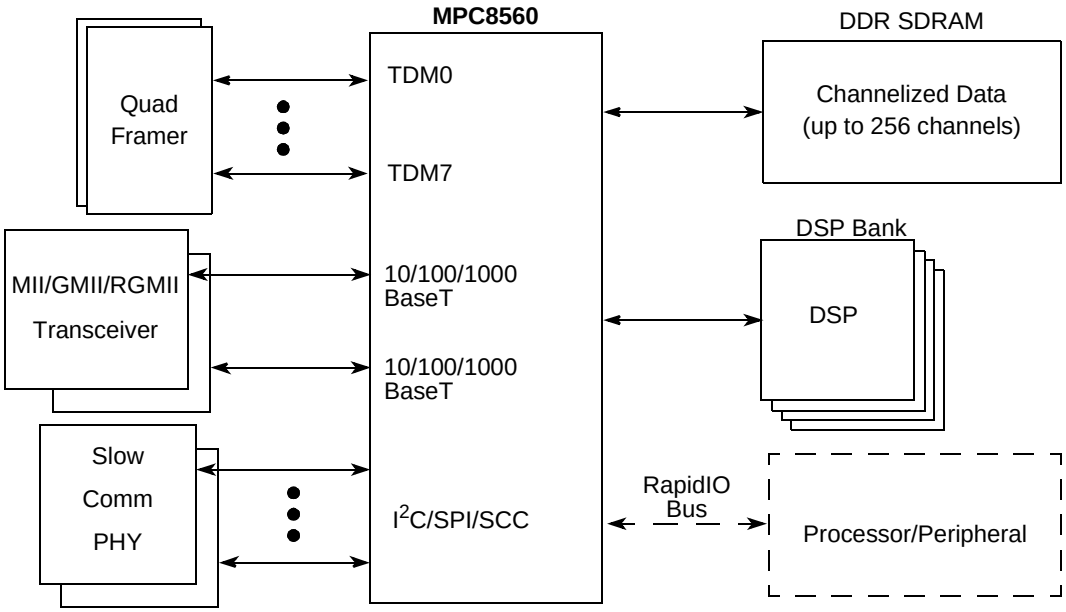


Figure 1-11. Cellular Base Station Configuration

Here the MPC8560 channelizes 8 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.

1.5.2.5 3G Wireless Base Station

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

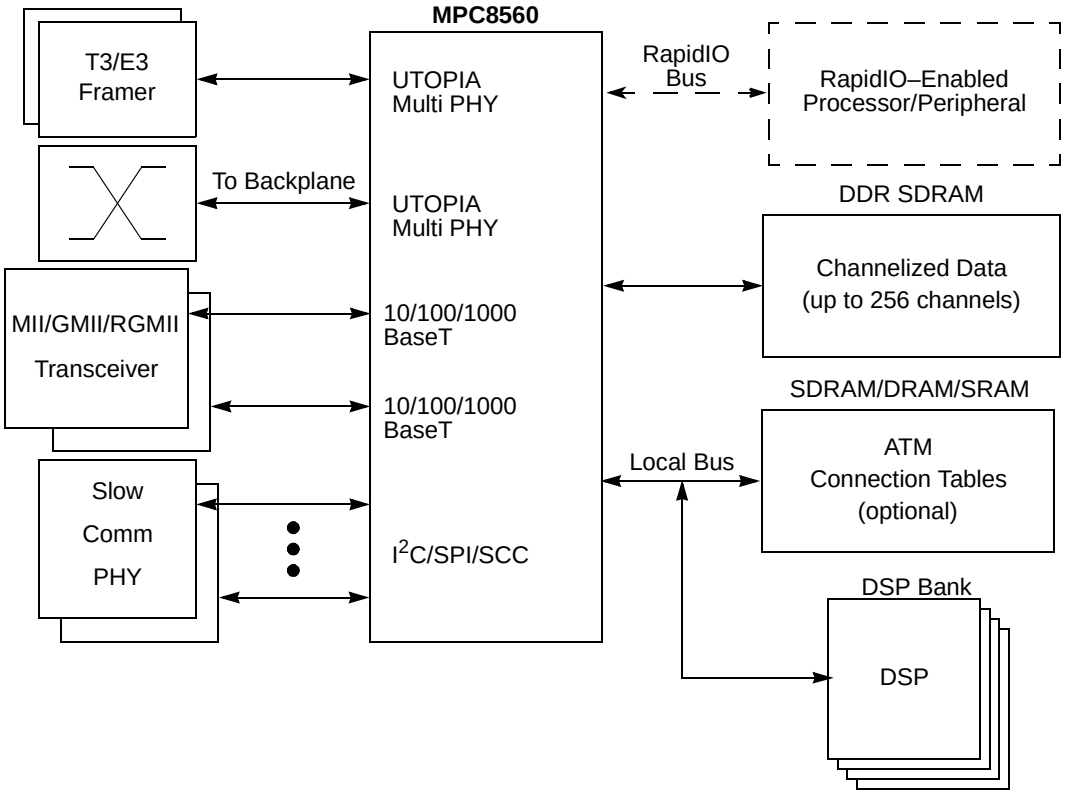


Figure 1-12. 3G Wireless Base Station Configuration

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.

1.5.2.6 Telecommunications Switch Controller

Figure 1-13 shows a telecommunications switch controller configuration.

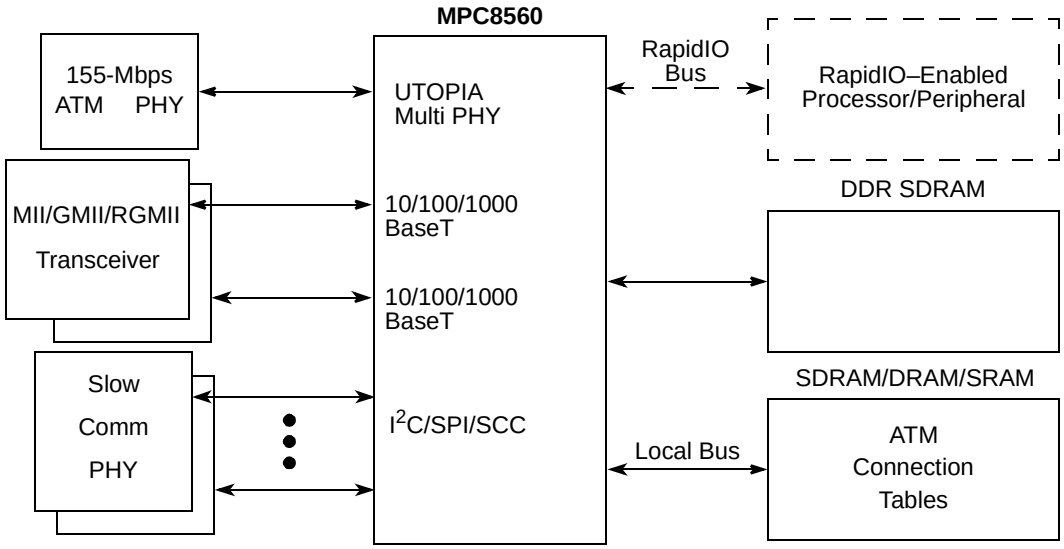


Figure 1-13. Telecommunications Switch Controller Configuration

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

1.5.2.7 SONET Transmission Controller

Figure 1-14 shows a SONET transmission controller configuration.

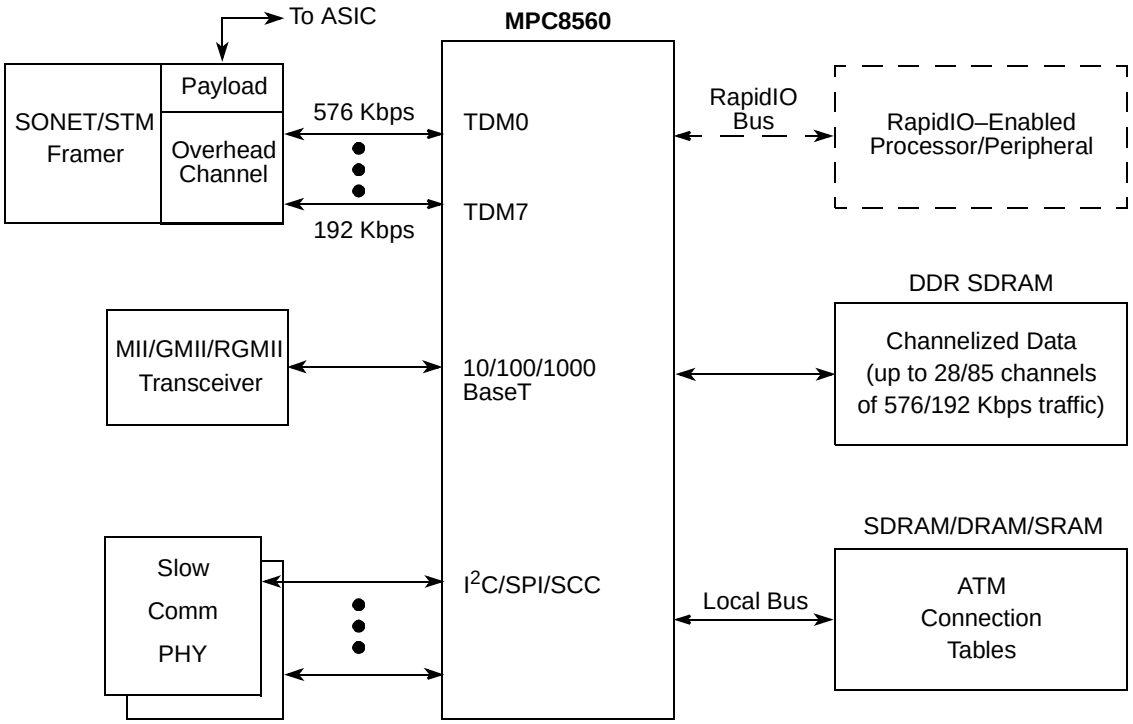


Figure 1-14. SONET Transmission Controller Configuration

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

1.5.2.8 Frame Relay Card

Figure 1-15 shows a frame relay card configuration.

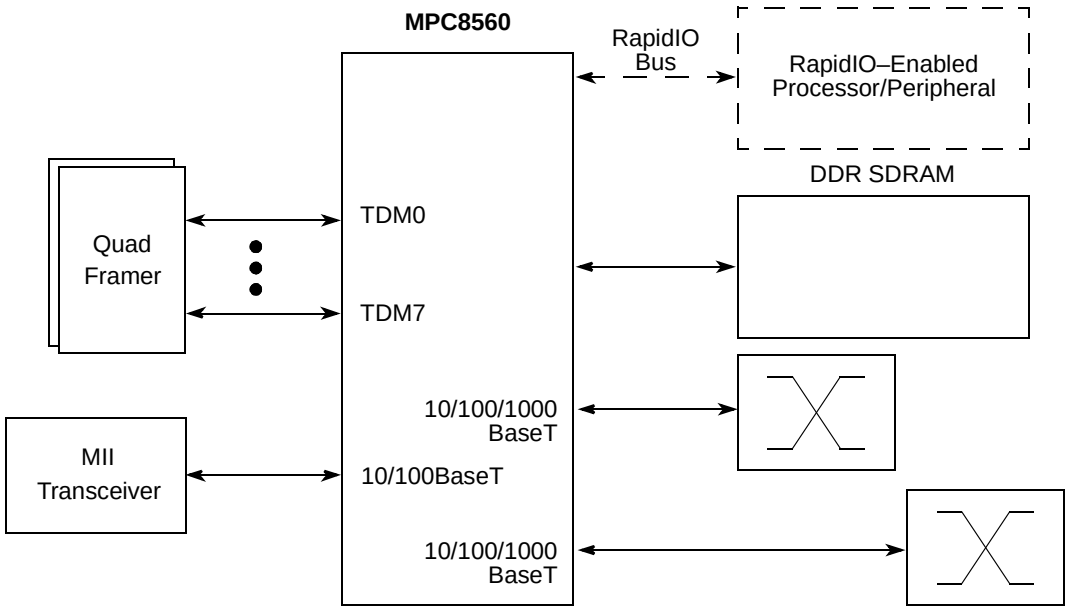


Figure 1-15. Frame Relay Card Configuration

1.5.2.9 ATM Protocol Converter

Figure 1-16 shows an ATM protocol converter.

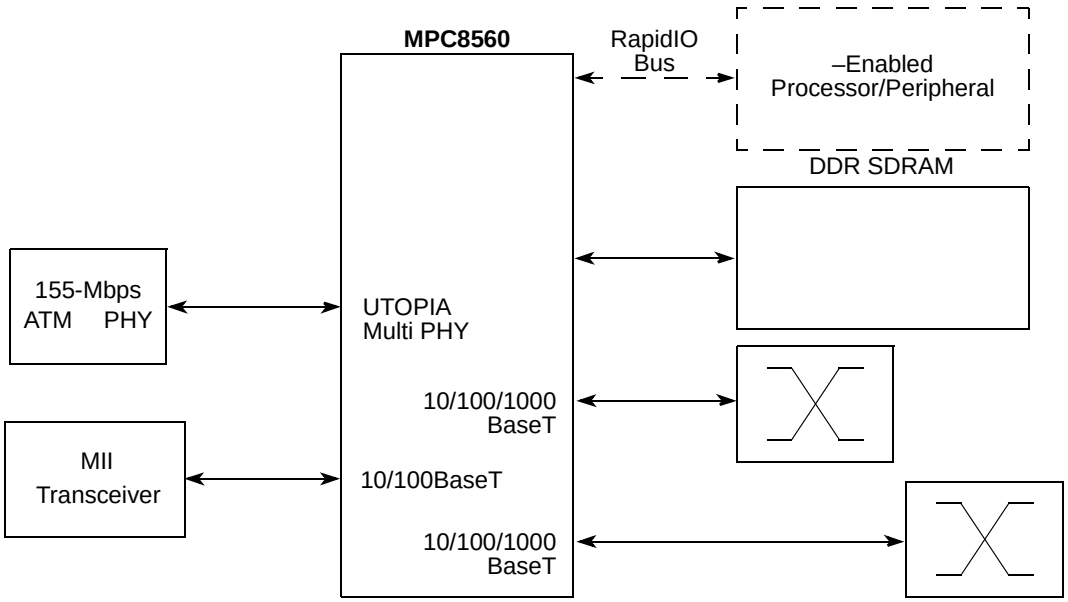


Figure 1-16. ATM Protocol Converter Configuration

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 through the two TSECs.

1.6 Compatibility Issues

This section describes some software and hardware compatibility issues.

1.6.1 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 (Freescale Semiconductor will provide the reference code.)

1.6.2 MPC8560 Hardware

As the MPC8560 family migrates to smaller geometries, the core voltage will reduce from 1.2 V to lower voltages. A programmable voltage regulator is recommended for future compatibility. See *MPC8560 Integrated Processor Hardware Specifications* for the electrical requirements and the AC and DC characteristics.

1.6.3 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

1.6.4 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				√	

1.6.5 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.

1.6.6 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.

1.6.7 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, such as 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: 100BaseT</td><td colspan="3">3 x 100BaseT 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 MPC8560 memory map. An overview of the local address map is followed by a description of how local access windows are used to define the local address map. The inbound and outbound address translation mechanisms used to map to and from external memory spaces are described next. Finally, the configuration, control, and status registers are described, including a complete listing of all memory-mapped registers with cross references to the sections detailing descriptions of each.

2.1 Local Memory Map Overview and Example

The 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. DMA engines also see this same local memory map. All memory accessed by the MPC8560 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 map 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 shows an example memory map.

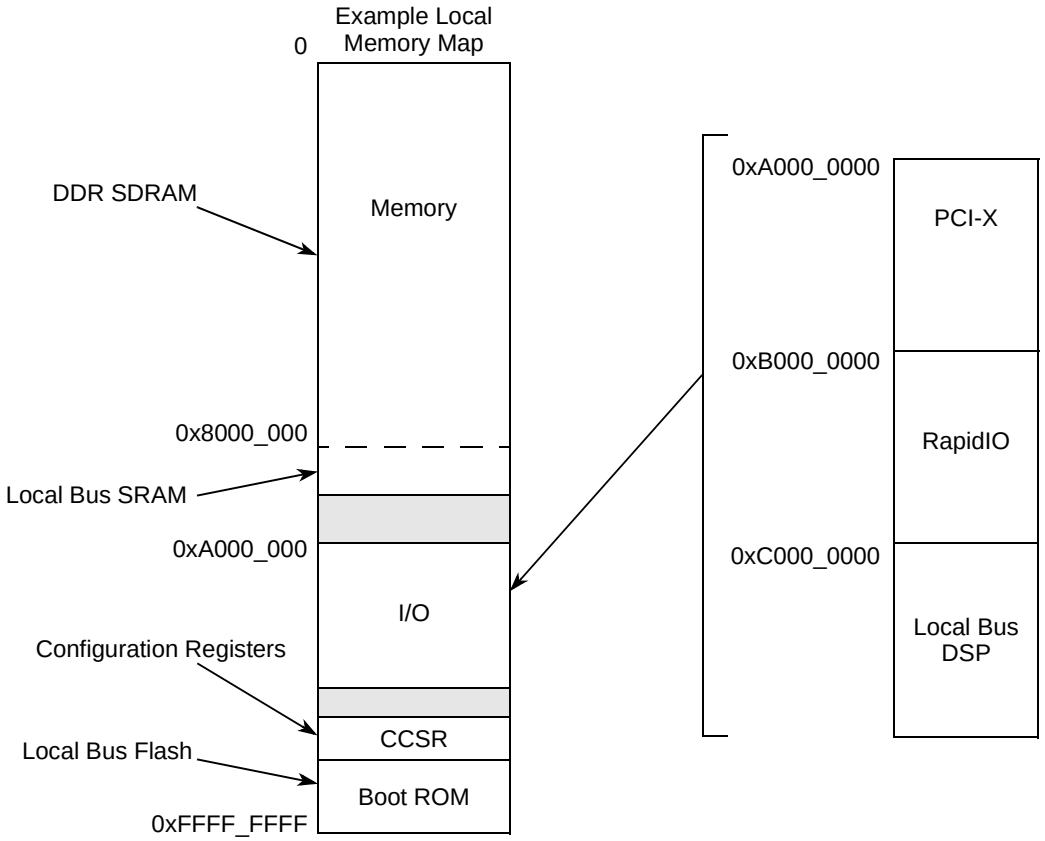


Figure 2-1. Local Memory Map Example

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

Table 2-2. Local Access Windows Example

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

In this example, it is not necessary to use a local access window to specify the location of the boot ROM because it is in the default location at the highest 8 Mbytes of memory (see Section 4.4.3.3, “Boot ROM Location”). 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 CCSRBAR. See Section 4.3.1.1.2, “Configuration, Control, and Status Base Address Register

(CCSRBAR).” However, note that the e500 core only provides one default TLB entry to access boot code and it allows for accesses within the highest 4 Kbytes of memory. In order to access the full 8 Mbytes of default boot space (and the 1 Mbyte of CCSR space), additional TLB entries must be set up within the e500 core for mapping these regions.

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 supersede all other mappings of these addresses for processor and global (snoopable) I/O transactions. Therefore, SRAM windows must never overlap configuration space as defined by CCSRBAR. It is possible to have SRAM windows overlap local access windows, but this is discouraged because processor and snoopable I/O transactions would map to the SRAM while non-snooped I/O transactions would be mapped by the local access windows. Only if all accesses to the SRAM address range are snoopable can results be consistent if the SRAM window overlaps a local access window.

See Section 7.3.1.4, “L2 Memory-Mapped SRAM Base Address Registers 0–1 (L2SRBARn),” for information about configuring SRAM windows.

2.2.2 Window into Configuration Space

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

2.2.3 Local Access Windows

As demonstrated in the address map overview in Section 2.1, “Local Memory Map Overview and Example,” local access windows associate a range of the local 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 in the following sections. Note that the minimum size of a window is 4 Kbytes, so the low order 12 bits of the base address cannot be specified.

2.2.3.1 Local Access Register Memory Map

Table 2-4 shows the memory map for the local access registers.

Table 2-4. Local Access Register Memory Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0C08	LAWBAR0—Local access window 0 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C10	LAWAR0—Local access window 0 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C28	LAWBAR1—Local access window 1 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C30	LAWAR1—Local access window 1 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C48	LAWBAR2—Local access window 2 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C50	LAWAR2—Local access window 2 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C68	LAWBAR3—Local access window 3 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C70	LAWAR3—Local access window 3 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C88	LAWBAR4—Local access window 4 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C90	LAWAR4—Local access window 4 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0CA8	LAWBAR5—Local access window 5 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0CB0	LAWAR5—Local access window 5 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0CC8	LAWBAR6—Local access window 6 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0CD0	LAWAR6—Local access window 6 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0CE8	LAWBAR7—Local access window 7 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0CF0	LAWAR7—Local access window 7 attribute register	R/W	0x0000_0000	2.2.3.3/2-6

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

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

	0										11	12	31
R	0	0	0	0	0	0	0	0	0	0	0	BASE_ADDR	
W													
Reset	0000_0000_0000_0000_0000_0000_0000_0000												
Offset	0xC08, 0xC28, 0xC48, 0xC68, 0xC88, 0xCA8, 0xCC8, 0xCE8												

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

Table 2-5 describes LAWBAR*n* field descriptions.

Table 2-5. LAWBAR*n* Field Descriptions

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 *n*. The specified base address should be aligned to the window size, as defined by LAWAR*n*[SIZE].

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

Figure 2-3 shows the bit fields of the LAWAR*n* registers.

	0	1					7	8	11	12											25	26	31
R	EN	0	0	0	0	0	0	0	TRGT_IF	0	0	0	0	0	0	0	0	0	0	0	0	0	SIZE
W																							
Reset	0000_0000_0000_0000_0000_0000_0000_0000																						
Offset	0xC10, 0xC30, 0xC50, 0xC70, 0xC90, 0xCB0, 0xCD0, 0xCF0																						

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

Table 2-6 describes LAWAR*n* field descriptions.

Table 2-6. LAWAR*n* Field Descriptions

Bits	Name	Description
0	EN	0 The local access window *n* (and all other LAWAR*n* and LAWBAR*n* fields) are disabled. 1 The local access window *n* is enabled and other LAWAR*n* and LAWBAR*n* fields combine to identify an address range for this window.
1–7	—	Write reserved, read = 0

Table 2-6. LAWAR*n* Field Descriptions (continued)

Bits	Name	Description
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 CCSRBAR and L2SRBAR. These mappings supersede local access window mappings, so configuration registers and SRAM do not appear as a target for local access windows. 0000 PCI/PCI-X 0001–0011 Reserved 0100 Local bus memory controller 0101–1011 Reserved 1100 RapidIO 1101–1110 Reserved 1111 DDR SDRAM
12–25	—	Write reserved, read = 0
26–31	SIZE	Identifies the size of the window from the starting address. Window size is $2^{(SIZE+1)}$ bytes. 000000–001010 Reserved 001011 4 Kbytes 001100 8 Kbytes 001101 16 Kbytes . . . . . $2^{(SIZE+1)}$ bytes 011110 2 Gbytes 011111–111111 Reserved

2.2.3.4 Precedence of Local Access Windows

If two local access windows overlap, the lower numbered window takes precedence. For instance, if two windows are setup 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
1	0x7FF0_0000	1 Mbyte	0b0100 (Local bus controller —LBC)
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 internal interconnects from the transactions source to its target. After the transaction has arrived at its target interface, that interface controller may perform additional mapping. For instance, the DDR SDRAM 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 a beginning and ending address, it is easy to define them so that they do not overlap with local access windows that map to other interfaces.

2.2.4 Outbound Address Translation and Mapping Windows

Outbound address translation and mapping refers to the translation of addresses from the local 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 16.3.2.2, “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 15.3.1.2, “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 16.3.2.2, “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 15.3.1.3, “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 4.3.1.1.2, "Configuration, Control, and Status 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.

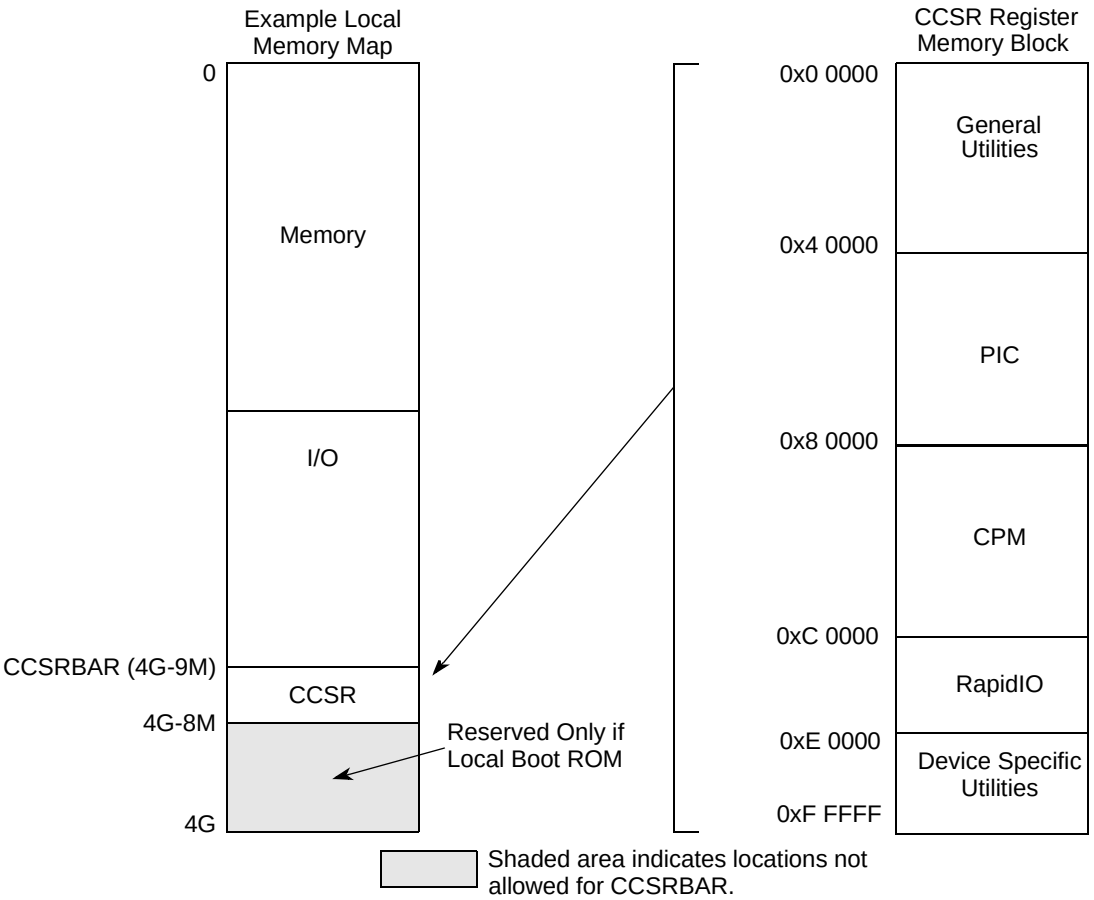


Figure 2-4. Top-Level Register Map Example

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 15.3.2.11, "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, see Section 16.3.1.12, "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 for a 4-Kbyte address space for registers. Registers that fall into this category are referred to as general utilities registers. These registers occupy the first 256 Kbytes of CCSR memory.

Registers 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 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

Figure 2-5 provides an overview of the general utilities registers.

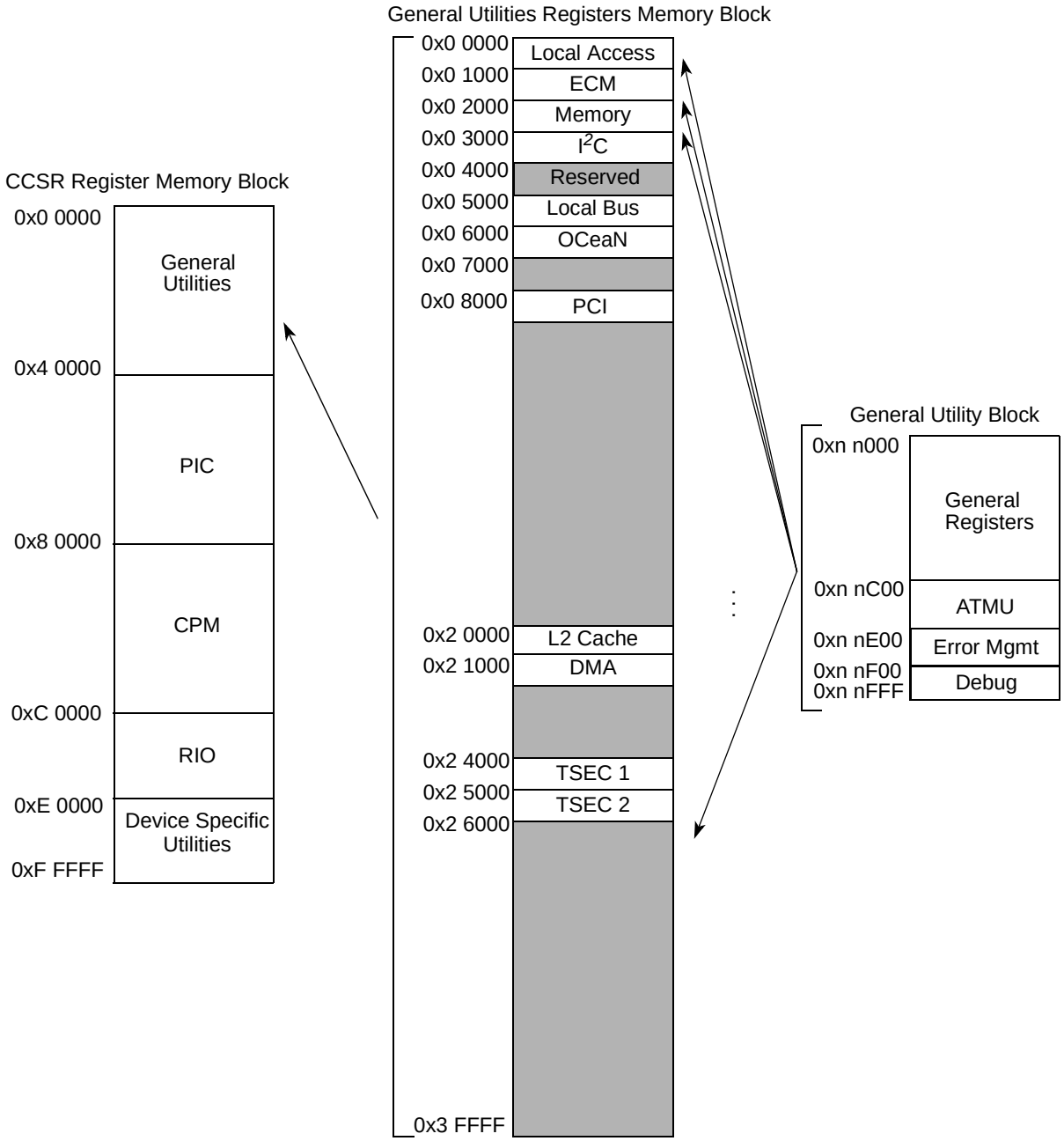


Figure 2-5. General Utilities Registers Mapping to Configuration, Control, and Status Memory Block

Figure 2-5 also shows the organization of registers inside the 4-Kbyte register space allocated to an individual functional block. The first 3 Kbytes are available for general registers. The next 512 bytes are dedicated to address translation and mapping registers, if applicable to that particular

functional unit (for example, PCI). 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, see Figure 2-6. Its programming model follows the OpenPIC architecture. The interrupt controller registers should only be accessed with 32-bit accesses.

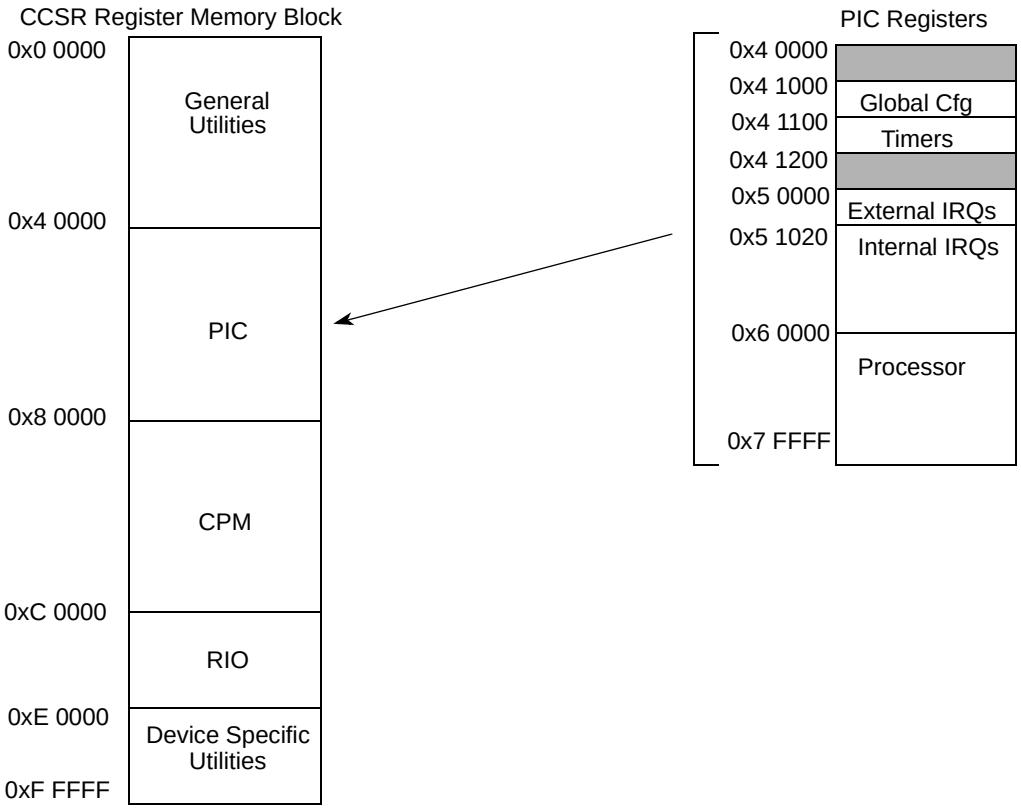


Figure 2-6. PIC Mapping to Configuration, Control, and Status Memory Block

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.

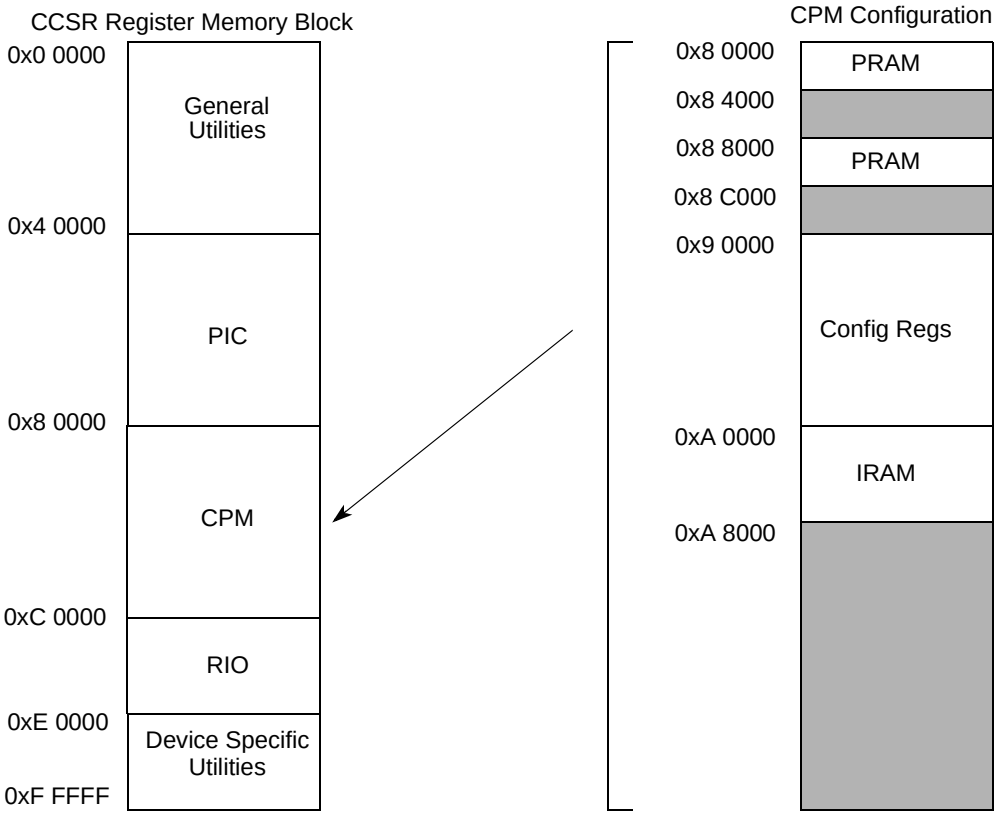


Figure 2-7. CPM Mapping to Configuration, Control, and Status Memory Block

2.3.6 RapidIO and CCSR

The RapidIO module uses 128 Kbytes of CCSR memory; refer to Figure 2-8. 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.

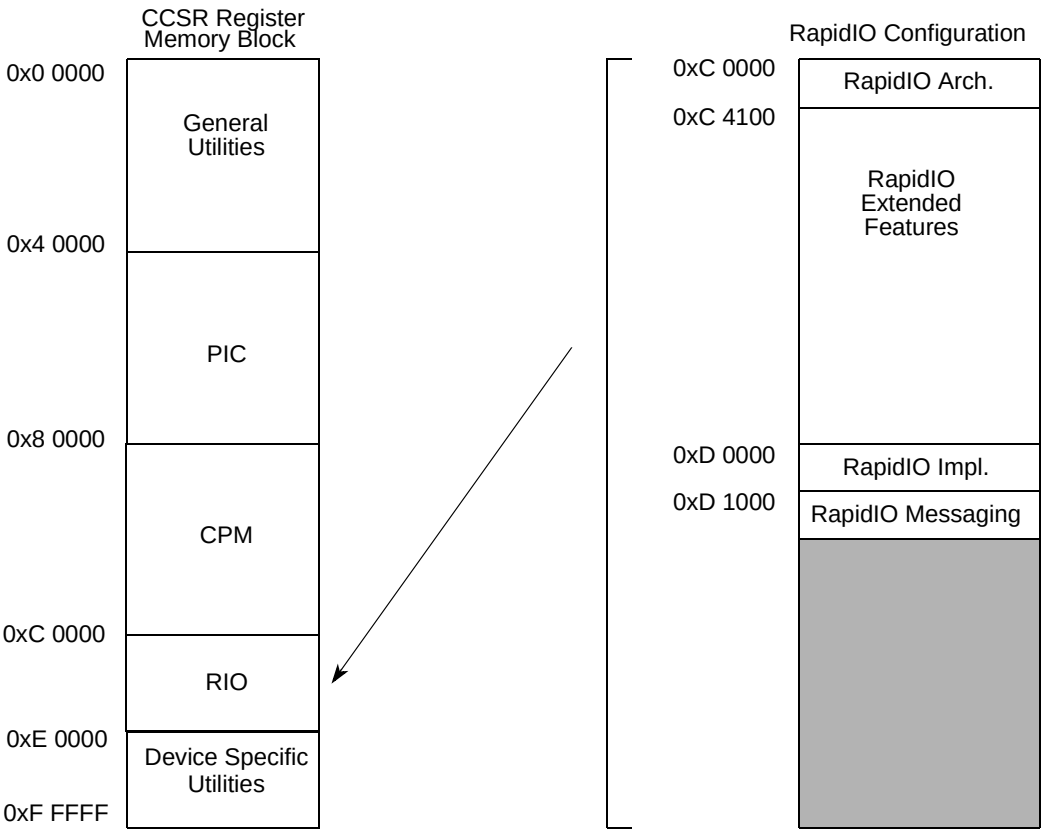


Figure 2-8. RapidIO Mapping to Configuration, Control, and Status Memory Block

2.3.7 Device-Specific Utilities

The device-specific registers consist of power management, performance monitors, and device-wide debug utilities (refer to Figure 2-9). These registers are accessible with 32-bit accesses only. Transactions of other than 32-bit are considered a programming error and operation is undefined.

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

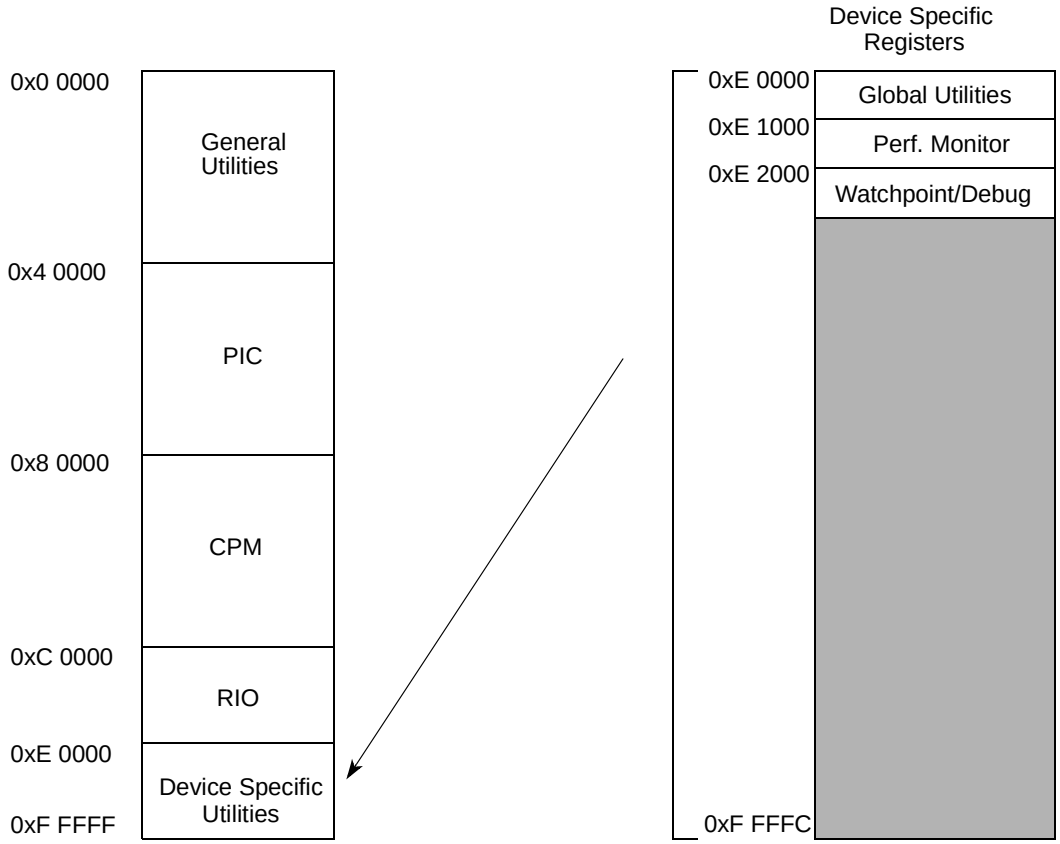


Figure 2-9. Device-Specific Register Mapping to Configuration, Control, and Status Memory Block

2.4 Complete CCSR Map

Table 2-9 lists the MPC8560 memory-mapped registers.

Table 2-9. Memory Map

Offset	Register	Access	Reset	Section/Page
Local-Access Registers—Configuration, Control, and Status Registers				
0x0_0000	CCSRBAR—Configuration, control, and status registers base address register	R/W	0x000F_F700	4.3.1.1.2/4-5
0x0_0008	ALTCBAR—Alternate configuration base address register	R/W	0x0000_0000	4.3.1.2.1/4-6
0x0_0010	ALTCAR—Alternate configuration attribute register	R/W	0x0000_0000	4.3.1.2.2/4-6
0x0_0020	BPTR—Boot page translation register	R/W	0x0000_0000	4.3.1.3.1/4-8

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
Local-Access Registers—Local-Access Window Base and Size Registers				
0x0_0C08	LAWBAR0—Local access window 0 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C10	LAWAR0—Local access window 0 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C28	LAWBAR1—Local access window 1 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C30	LAWAR1—Local access window 1 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C48	LAWBAR2—Local access window 2 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C50	LAWAR2—Local access window 2 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C68	LAWBAR3—Local access window 3 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C70	LAWAR3—Local access window 3 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0C88	LAWBAR4—Local access window 4 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0C90	LAWAR4—Local access window 4 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0CA8	LAWBAR5—Local access window 5 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0CB0	LAWAR5—Local access window 5 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0CC8	LAWBAR6—Local access window 6 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0CD0	LAWAR6—Local access window 6 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
0x0_0CE8	LAWBAR7—Local access window 7 base address register	R/W	0x0000_0000	2.2.3.2/2-6
0x0_0CF0	LAWAR7—Local access window 7 attribute register	R/W	0x0000_0000	2.2.3.3/2-6
Registers				
0x0_1000	EEBACR—ECM CCB address configuration register	R/W	0x0000_0003	8.2.1.1/8-3
0x0_1010	EEBPCR—ECM CCB port configuration register	R/W	0x0*n*00_0000	8.2.1.2/8-4
0x0_1E00	EEDR—ECM error detect register	R/W	0x0000_0000	8.2.1.3/8-5
0x0_1E08	EEER—ECM error enable register	R/W	0x0000_0000	8.2.1.4/8-6
0x0_1E0C	EEATR—ECM error attributes capture register	R	0x0000_0000	8.2.1.5/8-7
0x0_1E10	EEADR—ECM error address capture register	R	0x0000_0000	8.2.1.6/8-8

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
DDR Memory Controller Memory Map				
0x0_2000	CS0_BNDS—Chip select 0 memory bounds	R/W	0x0000_0000	9.4.1.1/9-10
0x0_2008	CS1_BNDS—Chip select 1 memory bounds			
0x0_2010	CS2_BNDS—Chip select 2 memory bounds			
0x0_2018	CS3_BNDS—Chip select 3 memory bounds			
0x0_2080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	9.4.1.2/9-10
0x0_2084	CS1_CONFIG—Chip select 1 configuration			
0x0_2088	CS2_CONFIG—Chip select 2 configuration			
0x0_208C	CS3_CONFIG—Chip select 3 configuration			
0x0_2108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	9.4.1.3/9-11
0x0_210C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	9.4.1.4/9-13
0x0_2110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.5/9-14
0x0_2118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	9.4.1.6/9-16
0x0_2124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	9.4.1.7/9-16
0x0_2E00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	9.4.1.8/9-17
0x0_2E04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	9.4.1.9/9-18
0x0_2E08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	9.4.1.10/9-18
0x0_2E20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	9.4.1.11/9-19
0x0_2E24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	9.4.1.12/9-20
0x0_2E28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	9.4.1.13/9-20
0x0_2E40	ERR_DETECT—Memory error detect	R/W	0x0000_0000	9.4.1.14/9-21
0x0_2E44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	9.4.1.15/9-21
0x0_2E48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	9.4.1.16/9-22
0x0_2E4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	9.4.1.17/9-23
0x0_2E50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	9.4.1.18/9-24
0x0_2E58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	9.4.1.19/9-25

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
I^2C				
0x0_3000	I2CADR—I^2C address register	R/W	0x00	11.3.1.1/11-5
0x0_3004	I2CFDR—I^2C frequency divider register	R/W	0x00	11.3.1.2/11-5
0x0_3008	I2CCR—I^2C control register	R/W	0x00	11.3.1.3/11-6
0x0_300C	I2CSR—I^2C status register	R/W	0x81	11.3.1.4/11-7
0x0_3010	I2CDR—I^2C data register	R/W	0x00	11.3.1.5/11-9
0x0_3014	I2CDFSRR—I^2C digital filter sampling rate register	R/W	0x10	11.3.1.6/11-10
Local Bus Controller Registers				
0x0_5000	BR0—Base register 0	R/W	0x0000_*nn*01[1]	12.3.1.1/12-10
0x0_5008	BR1—Base register 1		0x0000_0000	
0x0_5010	BR2—Base register 2			
0x0_5018	BR3—Base register 3			
0x0_5020	BR4—Base register 4			
0x0_5028	BR5—Base register 5			
0x0_5030	BR6—Base register 6			
0x0_5038	BR7—Base register 7			
0x0_5004	OR0—Options register 0	R/W	0x0000_0FF7	12.3.1.2/12-11
0x0_500C	OR1—Options register 1		0x0000_0000	
0x0_5014	OR2—Options register 2			
0x0_501C	OR3—Options register 3			
0x0_5024	OR4—Options register 4			
0x0_502C	OR5—Options register 5			
0x0_5034	OR6—Options register 6			
0x0_503C	OR7—Options register 7			
0x0_5068	MAR—UPM address register	R/W	0x0000_0000	12.3.1.3/12-17
0x0_5070	MAMR—UPMA mode register	R/W	0x0000_0000	12.3.1.4/12-18
0x0_5074	MBMR—UPMB mode register	R/W	0x0000_0000	12.3.1.4/12-18
0x0_5078	MCMR—UPMC mode register	R/W	0x0000_0000	12.3.1.4/12-18
0x0_5084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	12.3.1.5/12-20
0x0_5088	MDR—UPM data register	R/W	0x0000_0000	12.3.1.6/12-21
0x0_5094	LSDMR—SDRAM mode register	R/W	0x0000_0000	12.3.1.7/12-21
0x0_50A0	LURT—UPM refresh timer	R/W	0x0000_0000	12.3.1.8/12-23

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_50A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	12.3.1.9/12-24
0x0_50B0	LTESR—Transfer error status register	Read/ Bit-reset	0x0000_0000	12.3.1.10/12-25
0x0_50B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	12.3.1.11/12-26
0x0_50B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	12.3.1.12/12-27
0x0_50BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	12.3.1.13/12-28
0x0_50C0	LTEAR—Transfer error address register	R/W	0x0000_0000	12.3.1.14/12-29
0x0_50D0	LBCR—Configuration register	R/W	0x0000_0000	12.3.1.15/12-30
0x0_50D4	LCRR—Clock ratio register	R/W	0x8000_0008	12.3.1.16/12-31
PCI/X Registers				
PCI/X Configuration Access Registers				
0x0_8000	CFG_ADDR—PCI/X configuration address	R/W	0x0000_0000	15.3.1.1.1/15-18
0x0_8004	CFG_DATA—PCI/X configuration data	R/W	0x0000_0000	15.3.1.1.1/15-18
0x0_8008	INT_ACK—PCI/X interrupt acknowledge	R	0x0000_0000	15.3.1.1.3/15-20
0x0_800C–0x0_8BFC	Reserved	—	—	—
PCI/X ATMU Registers—Outbound and Inbound				
0x0_8C00–0x0_8C3C–Outbound Window 0 (default)				
0x0_8C00	POTAR0—PCI/X outbound window 0 (default) translation address register	R/W	0x0000_0000	15.3.1.2.1/15-21
0x0_8C04	POTEAR0—PCI/X outbound window 0 (default) translation extended address register	R/W	0x0000_0000	15.3.1.2.2/15-21
0x0_8C08	Reserved	—	—	
0x0_8C0C	Reserved	—	—	
0x0_8C10	POWAR0—PCI/X outbound window 0 (default) attributes register	R/W	0x8004_401F	15.3.1.2.4/15-22
0x0_8C14–0x0_8C1C	Reserved	—	—	
0x0_8C20–0x0_8C3C—Outbound Window 1				
0x0_8C20	POTAR1—PCI/X outbound window 1 translation address register	R/W	0x0000_0000	15.3.1.2.1/15-21
0x0_8C24	POTEAR1—PCI/X outbound window 1 translation extended address register	R/W	0x0000_0000	15.3.1.2.2/15-21
0x0_8C28	POWBAR1—PCI/X outbound window 1 base address register	R/W	0x0000_0000	15.3.1.2.3/15-22

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8C2C	Reserved	—	—	
0x0_8C30	POWAR1—PCI/X outbound window 1 attributes register	R/W	0x0000_0000	15.3.1.2.4/15-22
0x0_8C34–0x0_8C3C	Reserved	—	—	
0x0_8C40–0x0_8C5C—Outbound Window 2				
0x0_8C40	POTAR2—PCI/X outbound window 2 translation address register	R/W	0x0000_0000	15.3.1.2.1/15-21
0x0_8C44	POTEAR2—PCI/X outbound window 2 translation extended address register	R/W	0x0000_0000	15.3.1.2.2/15-21
0x0_8C48	POWBAR2—PCI/X outbound window 2 base address register	R/W	0x0000_0000	15.3.1.2.3/15-22
0x0_8C4C	Reserved	—	—	
0x0_8C50	POWAR2—PCI/X outbound window 2 attributes register	R/W	0x0000_0000	15.3.1.2.4/15-22
0x0_8C54–0x0_8C5C	Reserved	—	—	
0x0_8C60–0x0_8C7C—Outbound Window 3				
0x0_8C60	POTAR3—PCI/X outbound window 3 translation address register	R/W	0x0000_0000	15.3.1.2.1/15-21
0x0_8C64	POTEAR3—PCI/X outbound window 3 translation extended address register	R/W	0x0000_0000	15.3.1.2.2/15-21
0x0_8C68	POWBAR3—PCI/X outbound window 3 base address register	R/W	0x0000_0000	15.3.1.2.3/15-22
0x0_8C6C	Reserved	—	—	
0x0_8C70	POWAR3—PCI/X outbound window 3 attributes register	R/W	0x0000_0000	15.3.1.2.4/15-22
0x0_8C74–0x0_8C7C	Reserved	—	—	
0x0_8C80–0x0_8C9C—Outbound Window 4				
0x0_8C80	POTAR4—PCI/X outbound window 4 translation address register	R/W	0x0000_0000	15.3.1.2.1/15-21
0x0_8C84	POTEAR4—PCI/X outbound window 4 translation extended address register	R/W	0x0000_0000	15.3.1.2.2/15-21
0x0_8C88	POWBAR4—PCI/X outbound window 4 base address register	R/W	0x0000_0000	15.3.1.2.3/15-22
0x0_8C8C	Reserved	—	—	

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8C90	POWAR4—PCI/X outbound window 4 attributes register	R/W	0x0000_0000	15.3.1.2.4/15-22
0x0_8C94–0x0_8D9C	Reserved	—	—	
0x0_8DA0–0x0_8DBC–Inbound Window 3				
0x0_8DA0	PITAR3—PCI/X inbound window 3 translation address register	R/W	0x0000_0000	15.3.1.3.1/15-25
0x0_8DA4	Reserved	—	—	
0x0_8DA8	PIWBAR3—PCI/X inbound window 3 base address register	R/W	0x0000_0000	15.3.1.3.2/15-25
0x0_8DAC	PIWBEAR3—PCI/X inbound window 3 base extended address register	R/W	0x0000_0000	15.3.1.3.3/15-26
0x0_8DB0	PIWAR3—PCI/X inbound window 3 attributes register	R/W	0x0000_0000	15.3.1.3.4/15-26
0x0_8DB4–0x0_8DBC	Reserved	—	—	
0x0_8DC0–0x0_8DDC–Inbound Window 2				
0x0_8DC0	PITAR2—PCI/X inbound window 2 translation address register	R/W	0x0000_0000	15.3.1.3.1/15-25
0x0_8DC4	Reserved	—	—	
0x0_8DC8	PIWBAR2—PCI/X inbound window 2 base address register	R/W	0x0000_0000	15.3.1.3.2/15-25
0x0_8DCC	PIWBEAR2—PCI/X inbound window 2 base extended address register	R/W	0x0000_0000	15.3.1.3.3/15-26
0x0_8DD0	PIWAR2—PCI/X inbound window 2 attributes register	R/W	0x0000_0000	15.3.1.3.4/15-26
0x0_8DD4–0x0_8DDC	Reserved	—	—	
0x0_8DE0–0x0_8DFC–Inbound Window 1				
0x0_8DE0	PITAR1—PCI/X inbound window 1 translation address register	R/W	0x0000_0000	15.3.1.3.1/15-25
0x0_8DE4	Reserved	—	—	
0x0_8DE8	PIWBAR1—PCI/X inbound window 1 base address register	R/W	0x0000_0000	15.3.1.3.2/15-25
0x0_8DEC	Reserved	—	—	
0x0_8DF0	PIWAR1—PCI/X inbound window 1 attributes register	R/W	0x0000_0000	15.3.1.3.4/15-26
0x0_8DF4–0x0_8DFC	Reserved	—	—	

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
PCI/X Error Management Registers				
0x0_8E00	ERR_DR—PCI/X error detect register	Special	0x0000_0000	15.3.1.4.1/15-29
0x0_8E04	ERR_CAP_DR—PCI/X error capture disabled register	R/W	0x0000_0000	15.3.1.4.2/15-30
0x0_8E08	ERR_EN—PCI/X error enable register	R/W	0x0000_0000	15.3.1.4.3/15-31
0x0_8E0C	ERR_ATTRIB—PCI/X error attributes capture register	R/W	0x0000_0000	15.3.1.4.4/15-32
0x0_8E10	ERR_ADDR—PCI/X error address capture register	R/W	0x0000_0000	15.3.1.4.5/15-33
0x0_8E14	ERR_EXT_ADDR—PCI/X error extended address capture register	R/W	0x0000_0000	15.3.1.4.6/15-34
0x0_8E18	ERR_DL—PCI/X error data low capture register	R/W	0x0000_0000	15.3.1.4.7/15-34
0x0_8E1C	ERR_DH—PCI/X error data high capture register	R/W	0x0000_0000	15.3.1.4.8/15-34
0x0_8E20	GAS_TIMR—PCI/X gasket timer register	R/W	0x0000_0000	15.3.1.4.9/15-35
0x0_8E24	PCIX_TIMR—PCIX split completion timer register	R/W	0x0000_0000	15.3.1.4.10/15-36
0x0_8E28–0x0_8EFC	Reserved	—	—	
0x0_8F00–0x0_8FFC	Reserved for debug	—	—	
L2/SRAM Memory-Mapped Configuration Registers				
0x2_0000	L2CTL—L2 control register	R/W	0x2000_0000	7.3.1.1/7-7
0x2_0010	L2CEWAR0—L2 cache external write address register 0	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0018	L2CEWCR0—L2 cache external write control register 0	R/W	0x0000_0000	7.3.1.3/7-11
0x2_0020	L2CEWAR1—L2 cache external write address register 1	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0028	L2CEWCR1—L2 cache external write control register 1	R/W	0x0000_0000	7.3.1.3/7-11
0x2_0030	L2CEWAR2—L2 cache external write address register 2	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0038	L2CEWCR2—L2 cache external write control register 2	R/W	0x0000_0000	7.3.1.3/7-11
0x2_0040	L2CEWAR3—L2 cache external write address register 3	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0048	L2CEWCR3—L2 cache external write control register 3	R/W	0x0000_0000	7.3.1.3/7-11
0x2_0100	L2SRBAR0—L2 memory-mapped SRAM base address register 0	R/W	0x0000_0000	7.3.1.4/7-12
0x2_0108	L2SRBAR1—L2 memory-mapped SRAM base address register 1	R/W	0x0000_0000	7.3.1.4/7-12
0x2_0E00	L2ERRINJHI—L2 error injection mask high register	R/W	0x0000_0000	7.3.1.5.1/7-13
0x2_0E04	L2ERRINJLO—L2 error injection mask low register	R/W	0x0000_0000	7.3.1.5.1/7-13
0x2_0E08	L2ERRINJCTL—L2 error injection tag/ECC control register	R/W	0x0000_0000	7.3.1.5.1/7-13

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_0E20	L2CAPTDATAHI—L2 error data high capture register	R	0x0000_0000	7.3.1.5.2/7-15
0x2_0E24	L2CAPTDATALO—L2 error data low capture register	R	0x0000_0000	7.3.1.5.2/7-15
0x2_0E28	L2CAPTECC—L2 error syndrome register	R	0x0000_0000	7.3.1.5.2/7-15
0x2_0E40	L2ERRDET—L2 error detect register	Special	0x0000_0000	7.3.1.5.2/7-15
0x2_0E44	L2ERRDIS—L2 error disable register	R/W	0x0000_0000	7.3.1.5.2/7-15
0x2_0E48	L2ERRINTEN—L2 error interrupt enable register	R/W	0x0000_0000	7.3.1.5.2/7-15
0x2_0E4C	L2ERRATTR—L2 error attributes capture register	R/W	0x0000_0000	7.3.1.5.2/7-15
0x2_0E50	L2ERRADDR—L2 error address capture register	R	0x0000_0000	7.3.1.5.2/7-15
0x2_0E58	L2ERRCTL—L2 error control register	R/W	0x0000_0000	7.3.1.5.2/7-15
DMA Registers				
General Registers				
0x2_1100	MR*n*—DMA 0 mode register	R/W	0x0000_0000	14.3.2.1/14-10
0x2_1104	SR*n*—DMA 0 status register	Special	0x0000_0000	14.3.2.2/14-13
0x2_1108	Reserved	—	—	—
0x2_110C	CLNDAR*n*—DMA 0 current link descriptor address register	R/W	0x0000_0000	14.3.2.3/14-14
0x2_1110	SATR*n*—DMA 0 source attributes register	R/W	0x0000_0000	14.3.2.4/14-16
0x2_1114	SAR*n*—DMA 0 source address register	R/W	0x0000_0000	14.3.2.5/14-18
0x2_1118	DATR*n*—DMA 0 destination attributes register	R/W	0x0000_0000	14.3.2.6/14-19
0x2_111C	DAR*n*—DMA 0 destination address register	R/W	0x0000_0000	14.3.2.7/14-21
0x2_1120	BCR*n*—DMA 0 byte count register	R/W	0x0000_0000	14.3.2.8/14-23
0x2_1124	Reserved	—	—	—
0x2_1128	NLNDAR*n*—DMA 0 next link descriptor address register	R/W	0x0000_0000	14.3.2.9/14-23
0x2_1130	Reserved	—	—	—
0x2_1134	CLSDAR*n*—DMA 0 current list alternate base descriptor address register	R/W	0x0000_0000	14.3.2.10/14-24
0x2_1138	Reserved	—	—	—
0x2_113C	NLSDAR*n*—DMA 0 next list descriptor address register	R/W	0x0000_0000	14.3.2.11/14-25
0x2_1140	SSR*n*—DMA 0 source stride register	R/W	0x0000_0000	14.3.2.12/14-25
0x2_1144	DSR*n*—DMA 0 destination stride register	R/W	0x0000_0000	14.3.2.13/14-26
0x2_1148–0x2_117C	Reserved	—	—	—
0x2_1180	MR*n*—DMA 1 mode register	R/W	0x0000_0000	14.3.2.1/14-10

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_1184	SR*n*—DMA 1 status register	Special	0x0000_0000	14.3.2.2/14-13
0x2_1188	Reserved	—	—	—
0x2_118C	CLNDAR*n*—DMA 1 current link descriptor address register	R/W	0x0000_0000	14.3.2.3/14-14
0x2_1190	SATR*n*—DMA 1 source attributes register	R/W	0x0000_0000	14.3.2.4/14-16
0x2_1194	SAR*n*—DMA 1 source address register	R/W	0x0000_0000	14.3.2.5/14-18
0x2_1198	DATR*n*—DMA 1 destination attributes register	R/W	0x0000_0000	14.3.2.6/14-19
0x2_119C	DAR*n*—DMA 1 destination address register	R/W	0x0000_0000	14.3.2.7/14-21
0x2_11A0	BCR*n*—DMA 1 byte count register	R/W	0x0000_0000	14.3.2.8/14-23
0x2_11A4	Reserved	—	—	—
0x2_11A8	NLNDAR*n*—DMA 1 next link descriptor address register	R/W	0x0000_0000	14.3.2.9/14-23
0x2_11AC–0x2_11B0	Reserved	—	—	—
0x2_11B4	CLSDAR*n*—DMA 1 current list alternate base descriptor address register	R/W	0x0000_0000	14.3.2.10/14-24
0x2_11B8	Reserved	—	—	—
0x2_11BC	NLSDAR*n*—DMA 1 next list descriptor address register	R/W	0x0000_0000	14.3.2.11/14-25
0x2_11C0	SSR*n*—DMA 1 source stride register	R/W	0x0000_0000	14.3.2.12/14-25
0x2_11C4	DSR*n*—DMA 1 destination stride register	R/W	0x0000_0000	14.3.2.13/14-26
0x2_11C8–0x2_11FC	Reserved	—	—	—
0x2_1200	MR*n*—DMA 2 mode register	R/W	0x0000_0000	14.3.2.1/14-10
0x2_1204	SR*n*—DMA 2 status register	Special	0x0000_0000	14.3.2.2/14-13
0x2_1208	Reserved	—	—	—
0x2_120C	CLNDAR*n*—DMA 2 current link descriptor address register	R/W	0x0000_0000	14.3.2.3/14-14
0x2_1210	SATR*n*—DMA 2 source attributes register	R/W	0x0000_0000	14.3.2.4/14-16
0x2_1214	SAR*n*—DMA 2 source address register	R/W	0x0000_0000	14.3.2.5/14-18
0x2_1218	DATR*n*—DMA 2 destination attributes register	R/W	0x0000_0000	14.3.2.6/14-19
0x2_121C	DAR*n*—DMA 2 destination address register	R/W	0x0000_0000	14.3.2.7/14-21
0x2_1220	BCR*n*—DMA 2 byte count register	R/W	0x0000_0000	14.3.2.8/14-23
0x2_1224	Reserved	—	—	—
0x2_1228	NLNDAR*n*—DMA 2 next link descriptor address register	R/W	0x0000_0000	14.3.2.9/14-23
0x2_122C–0x2_1230	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_1234	CLSDAR*n*—DMA 2 current list alternate base descriptor address register	R/W	0x0000_0000	14.3.2.10/14-24
0x2_1238	Reserved	—	—	—
0x2_123C	NLSDAR*n*—DMA 2 next list descriptor address register	R/W	0x0000_0000	14.3.2.11/14-25
0x2_1240	SSR*n*—DMA 2 source stride register	R/W	0x0000_0000	14.3.2.12/14-25
0x2_1244	DSR*n*—DMA 2 destination stride register	R/W	0x0000_0000	14.3.2.13/14-26
0x2_1248–0x2_127C	Reserved	—	—	—
0x2_1280	MR*n*—DMA 3 mode register	R/W	0x0000_0000	14.3.2.1/14-10
0x2_1284	SR*n*—DMA 3 status register	Special	0x0000_0000	14.3.2.2/14-13
0x2_1288	Reserved	—	—	—
0x2_128C	CLNDAR*n*—DMA 3 current link descriptor address register	R/W	0x0000_0000	14.3.2.3/14-14
0x2_1290	SATR*n*—DMA 3 source attributes register	R/W	0x0000_0000	14.3.2.4/14-16
0x2_1294	SAR*n*—DMA 3 source address register	R/W	0x0000_0000	14.3.2.5/14-18
0x2_1298	DATR*n*—DMA 3 destination attributes register	R/W	0x0000_0000	14.3.2.6/14-19
0x2_129C	DAR*n*—DMA 3 destination address register	R/W	0x0000_0000	14.3.2.7/14-21
0x2_12A0	BCR*n*—DMA 3 byte count register	R/W	0x0000_0000	14.3.2.8/14-23
0x2_12A4	Reserved	—	—	—
0x2_12A8	NLNDAR*n*—DMA 3 next link descriptor address register	R/W	0x0000_0000	14.3.2.9/14-23
0x2_12AC–0x2_12B0	Reserved	—	—	—
0x2_12B4	CLSDAR*n*—DMA 3 current list alternate base descriptor address register	R/W	0x0000_0000	14.3.2.10/14-24
0x2_12B8	Reserved	—	—	—
0x2_12BC	NLSDAR*n*—DMA 3 next list descriptor address register	R/W	0x0000_0000	14.3.2.11/14-25
0x2_12C0	SSR*n*—DMA 3 source stride register	R/W	0x0000_0000	14.3.2.12/14-25
0x2_12C4	DSR*n*—DMA 3 destination stride register	R/W	0x0000_0000	14.3.2.13/14-26
0x2_12C8–0x2_12FC	Reserved	—	—	—
0x2_1300	DGSR—DMA general status register	Read	0x0000_0000	14.3.2.14/14-26
TSEC1 General Control and Status Registers				
0x2_4000–0x2_400C	Reserved	R	0x0000_0000	—
0x2_4010	IEVENT—Interrupt event register	R/W	0x0000_0000	13.5.3.1.1/13-20

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4014	IMASK—Interrupt mask register	R/W	0x0000_0000	13.5.3.1.2/13-23
0x2_4018	EDIS—Error disabled register	R/W	0x0000_0000	13.5.3.1.3/13-24
0x2_401C	Reserved	R	0x0000_0000	—
0x2_4020	ECNTRL—Ethernet control register	R/W	0x0000_0000	13.5.3.1.4/13-25
0x2_4024	MINFLR—Minimum frame length register	R/W	0x0000_0040	13.5.3.1.5/13-26
0x2_4028	PTV—Pause time value register	R/W	0x0000_0000	13.5.3.1.6/13-27
0x2_402C	DMACTRL—DMA control register	R/W	0x0000_0000	13.5.3.1.7/13-28
0x2_4030	TBIPA—TBI PHY address register	R/W	0x0000_0000	13.5.3.1.8/13-29
0x2_4034–0x2_4088	Reserved	R	0x0000_0000	—
TSEC1 FIFO Control and Status Registers				
0x2_404C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	13.5.3.2.1/13-30
0x2_4050–0x2_4088	Reserved	R	0x0000_0000	—
0x2_408C	FIFO_TX_THR—FIFO transmit threshold register	R/W	0x0000_0100	13.5.3.2.2/13-31
0x2_4090–0x2_4094	Reserved	R	0x0000_0000	—
0x2_4098	FIFO_TX_STARVE—FIFO transmit starve register	R/W	0x0000_0080	13.5.3.2.3/13-32
0x2_409C	FIFO_TX_STARVE_SHUTOFF—FIFO transmit starve shutoff register	R/W	0x0000_0100	13.5.3.2.4/13-32
0x2_40A0–0x2_40FC	Reserved	R	0x0000_0000	—
TSEC1 Transmit Control and Status Registers				
0x2_4100	TCTRL—Transmit control register	R/W	0x0000_0000	13.5.3.3.1/13-33
0x2_4104	TSTAT—Transmit status register	R/W	0x0000_0000	13.5.3.3.2/13-34
0x2_4108	Reserved	R	0x0000_0000	—
0x2_410C	TBDLEN—TxBD data length register	R	0x0000_0000	13.5.3.3.3/13-35
0x2_4110	TXIC—Transmit interrupt coalescing configuration register	R/W	0x0000_0000	13.5.3.3.4/13-35
0x2_4114–0x2_4120	Reserved	R	0x0000_0000	—
0x2_4124	CTBPTR—Current TxBD pointer register	R	0x0000_0000	13.5.3.3.5/13-36
0x2_4128–0x2_4180	Reserved	R	0x0000_0000	—
0x2_4184	TBPTR—TxBD pointer register	R/W	0x0000_0000	13.5.3.3.6/13-37

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4188–0x2_4200	Reserved	R	0x0000_0000	—
0x2_4204	TBASE—TxBD base address register	R/W	0x0000_0000	13.5.3.3.7/13-37
0x2_4208–0x2_42AC	Reserved	R	0x0000_0000	—
0x2_42B0	OSTBD—Out-of-sequence TxBD register	R/W	0x0800_0000	13.5.3.3.8/13-38
0x2_42B4	OSTBDP—Out-of-sequence Tx data buffer pointer register	R/W	0x0000_0000	13.5.3.3.9/13-40
0x2_42B8–0x2_42FC	Reserved	R	0x0000_0000	—
TSEC1 Receive Control and Status Registers				
0x2_4300	RCTRL—Receive control register	R/W	0x0000_0000	13.5.3.4.1/13-41
0x2_4304	RSTAT—Receive status register	R/W	0x0000_0000	13.5.3.4.2/13-41
0x2_4308	Reserved	R	0x0000_0000	—
0x2_430C	RBDLEN—RxBD data length register	R	0x0000_0000	13.5.3.4.3/13-42
0x2_4310	RXIC—Receive interrupt coalescing configuration register	R/W	0x0000_0000	13.5.3.4.4/13-43
0x2_4314–0x2_4320	Reserved	R	0x0000_0000	—
0x2_4324	CRBPTR—Current RxBD pointer register	R	0x0000_0000	13.5.3.4.5/13-44
0x2_4328–0x2_433C	Reserved	R	0x0000_0000	—
0x2_4340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	13.5.3.4.6/13-44
0x2_4344–0x2_4380	Reserved	R	0x0000_0000	—
0x2_4384	RBPTR—RxBD pointer register	R/W	0x0000_0000	13.5.3.4.7/13-45
0x2_4388–0x2_4400	Reserved	R	0x0000_0000	—
0x2_4404	RBASE—RxBD base address register	R/W	0x0000_0000	13.5.3.4.8/13-45
0x2_4408–0x2_44FC	Reserved	R	0x0000_0000	—
TSEC1 MAC Registers				
0x2_4500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	13.5.3.6.1/13-49
0x2_4504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	13.5.3.6.2/13-51
0x2_4508	IPGIFG—Inter-packet gap/inter-frame gap register	R/W	0x4060_5060	13.5.3.6.3/13-52
0x2_450C	HAFDUP—Half-duplex register	R/W	0x00A1_F037	13.5.3.6.4/13-53

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4510	MAXFRM—Maximum frame length register	R/W	0x0000_0600	13.5.3.6.5/13-54
0x2_4514–0x2_451C	Reserved	R	0x0000_0000	—
0x2_4520	MIIMCFG—MII management configuration register	R/W	0x0000_0000	13.5.3.6.6/13-54
0x2_4524	MIIMCOM—MII Management command register	R/W	0x0000_0000	13.5.3.6.7/13-55
0x2_4528	MIIMADD—MII Management address register	R/W	0x0000_0000	13.5.3.6.8/13-56
0x2_452C	MIIMCON—MII Management control register	W	0x0000_0000	13.5.3.6.9/13-57
0x2_4530	MIIMSTAT—MII Management status register	R	0x0000_0000	13.5.3.6.10/13-57
0x2_4534	MIIMIND—MII Management indicator register	R	0x0000_0000	13.5.3.6.11/13-58
0x2_4538	Reserved	R	0x0000_0000	—
0x2_453C	IFSTAT—Interface status register	R/W	0x0000_0000	13.5.3.6.12/13-58
0x2_4540	MACSTNADDR1—Station address register, part 1	R/W	0x0000_0000	13.5.3.6.13/13-59
0x2_4544	MACSTNADDR2—Station address register, part 2	R/W	0x0000_0000	13.5.3.6.14/13-60
0x2_4548–0x2_467C	Reserved	R	0x0000_0000	—
TSEC1 RMON MIB Registers				
TSEC1 Transmit and Receive Counters				
0x2_4680	TR64—Transmit and receive 64-byte frame counter register	R/W	0x0000_0000	13.5.3.7.1/13-61
0x2_4684	TR127—Transmit and receive 65- to 127-byte frame counter register	R/W	0x0000_0000	13.5.3.7.2/13-61
0x2_4688	TR255—Transmit and receive 128- to 255-byte frame counter register	R/W	0x0000_0000	13.5.3.7.3/13-62
0x2_468C	TR511—Transmit and receive 256- to 511-byte frame counter register	R/W	0x0000_0000	13.5.3.7.4/13-62
0x2_4690	TR1K—Transmit and receive 512- to 1023-byte frame counter register	R/W	0x0000_0000	13.5.3.7.5/13-63
0x2_4694	TRMAX—Transmit and receive 1024- to 1518-byte frame counter register	R/W	0x0000_0000	13.5.3.7.6/13-63
0x2_4698	TRMGV—Transmit and receive 1519- to 1522-byte good VLAN frame count register	R/W	0x0000_0000	13.5.3.7.7/13-64
TSEC1 Receive Counters				
0x2_469C	RBYT—receive byte counter register	R/W	0x0000_0000	13.5.3.7.8/13-64
0x2_46A0	RPKT—receive packet counter register	R/W	0x0000_0000	13.5.3.7.9/13-65
0x2_46A4	RFCS—receive FCS error counter register	R/W	0x0000_0000	13.5.3.7.10/13-65
0x2_46A8	RMCA—receive multicast packet counter register	R/W	0x0000_0000	13.5.3.7.11/13-66

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_46AC	RBCA—receive broadcast packet counter register	R/W	0x0000_0000	13.5.3.7.12/13-66
0x2_46B0	RXCF—receive control frame packet counter register	R/W	0x0000_0000	13.5.3.7.13/13-67
0x2_46B4	RXPF—receive PAUSE frame packet counter register	R/W	0x0000_0000	13.5.3.7.14/13-67
0x2_46B8	RXUO—receive unknown OP code counter register	R/W	0x0000_0000	13.5.3.7.15/13-68
0x2_46BC	RALN—receive alignment error counter register	R/W	0x0000_0000	13.5.3.7.16/13-68
0x2_46C0	RFLR—receive frame length error counter register	R/W	0x0000_0000	13.5.3.7.17/13-69
0x2_46C4	RCDE—receive code error counter register	R/W	0x0000_0000	13.5.3.7.18/13-69
0x2_46C8	RCSE—receive carrier sense error counter register	R/W	0x0000_0000	13.5.3.7.19/13-70
0x2_46CC	RUND—receive undersize packet counter register	R/W	0x0000_0000	13.5.3.7.20/13-70
0x2_46D0	ROVR—receive oversize packet counter register	R/W	0x0000_0000	13.5.3.7.21/13-71
0x2_46D4	RFRG—receive fragments counter register	R/W	0x0000_0000	13.5.3.7.22/13-71
0x2_46D8	RJBR—receive jabber counter register	R/W	0x0000_0000	13.5.3.7.23/13-72
0x2_46DC	RDRP—receive drop register	R/W	0x0000_0000	13.5.3.7.24/13-72
TSEC1 Transmit Counters				
0x2_46E0	TBYT—Transmit byte counter register	R/W	0x0000_0000	13.5.3.7.25/13-73
0x2_46E4	TPKT—Transmit packet counter register	R/W	0x0000_0000	13.5.3.7.26/13-73
0x2_46E8	TMCA—Transmit multicast packet counter register	R/W	0x0000_0000	13.5.3.7.27/13-74
0x2_46EC	TBCA—Transmit broadcast packet counter register	R/W	0x0000_0000	13.5.3.7.28/13-74
0x2_46F0	TXPF—Transmit PAUSE control frame counter register	R/W	0x0000_0000	13.5.3.7.29/13-75
0x2_46F4	TDFR—Transmit deferral packet counter register	R/W	0x0000_0000	13.5.3.7.30/13-75
0x2_46F8	TEDF—Transmit excessive deferral packet counter register	R/W	0x0000_0000	13.5.3.7.31/13-76
0x2_46FC	TSCL—Transmit single collision packet counter register	R/W	0x0000_0000	13.5.3.7.32/13-76
0x2_4700	TMCL—Transmit multiple collision packet counter register	R/W	0x0000_0000	13.5.3.7.33/13-77
0x2_4704	TLCL—Transmit late collision packet counter register	R/W	0x0000_0000	13.5.3.7.34/13-77
0x2_4708	TXCL—Transmit excessive collision packet counter register	R/W	0x0000_0000	13.5.3.7.35/13-78
0x2_470C	TNCL—Transmit total collision counter register	R/W	0x0000_0000	13.5.3.7.36/13-78
0x2_4710	Reserved	R	0x0000_0000	—
0x2_4714	TDRP—Transmit drop frame counter register	R/W	0x0000_0000	13.5.3.7.37/13-79
0x2_4718	TJBR—Transmit jabber frame counter register	R/W	0x0000_0000	13.5.3.7.38/13-79
0x2_471C	TFCS—Transmit FCS error counter register	R/W	0x0000_0000	13.5.3.7.39/13-80

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4720	TXCF—Transmit control frame counter register	R/W	0x0000_0000	13.5.3.7.40/13-80
0x2_4724	TOVR—Transmit oversize frame counter register	R/W	0x0000_0000	13.5.3.7.41/13-81
0x2_4728	TUND—Transmit undersize frame counter register	R/W	0x0000_0000	13.5.3.7.42/13-81
0x2_472C	TFRG—Transmit fragments frame counter register	R/W	0x0000_0000	13.5.3.7.43/13-82
TSEC1 General Registers				
0x2_4730	CAR1—Carry register one register	R/W	0x0000_0000	13.5.3.7.44/13-82
0x2_4734	CAR2—Carry register two register	R/W	0x0000_0000	13.5.3.7.45/13-84
0x2_4738	CAM1—Carry register one mask register	R/W	0xFE01_FFFF	13.5.3.7.46/13-85
0x2_473C	CAM2—Carry register two mask register	R/W	0x000F_FFFF	13.5.3.7.47/13-86
0x2_4740–0x2_47FC	Reserved	R	0x0000_0000	—
TSEC1 Hash Function Registers				
0x2_4800	IADDR0—Individual address register 0	R/W	0x0000_0000	13.5.3.8.1/13-88
0x2_4804	IADDR1—Individual address register 1	R/W	0x0000_0000	
0x2_4808	IADDR2—Individual address register 2	R/W	0x0000_0000	
0x2_480C	IADDR3—Individual address register 3	R/W	0x0000_0000	
0x2_4810	IADDR4—Individual address register 4	R/W	0x0000_0000	
0x2_4814	IADDR5—Individual address register 5	R/W	0x0000_0000	
0x2_4818	IADDR6—Individual address register 6	R/W	0x0000_0000	
0x2_481C	IADDR7—Individual address register 7	R/W	0x0000_0000	
0x2_4820–0x2_487C	Reserved	R	0x0000_0000	—
0x2_4880	GADDR0—Group address register 0	R/W	0x0000_0000	13.5.3.8.2/13-88
0x2_4884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x2_4888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x2_488C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x2_4890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x2_4894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x2_4898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x2_489C	GADDR7—Group address register 7	R/W	0x0000_0000	
0x2_48A0–0x2_4BF4	Reserved	R	0x0000_0000	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
TSEC1 Attribute Registers				
0x2_4BF8	ATTR—Attribute register	R/W	0x0000_0000	13.5.3.9.1/13-89
0x2_4BFC	ATTRELI—Attribute EL & EI register	R/W	0x0000_0000	13.5.3.9.2/13-90
0x2_48A0–0x2_4AFC	Reserved	R	0x0000_0000	—
TSEC1 Future Expansion Space				
0x2_4C00–0x2_4FFC	Reserved	R	0x0000_0000	—
0x2_5000–0x2_5FFC	TSEC2 registers[2]			
PIC Register Address Map—Global Registers				
0x4_0000–0x4_0030	Reserved	—	—	—
0x4_0040	IPIDR0—Interprocessor interrupt 0 (IPI 0) dispatch register	W	0x0000_0000	10.3.7.1/10-39
0x4_0050	IPIDR1—IPI 1 dispatch register			
0x4_0060	IPIDR2—IPI 2 dispatch register			
0x4_0070	IPIDR3—IPI 3 dispatch register			
0x4_0080	CTPR—Current task priority register	R/W	0x0000_000F	10.3.7.2/10-40
0x4_0090	WHOAMI—Who am I register	R	0x0000_0000	10.3.7.3/10-41
0x4_00A0	IACK—Interrupt acknowledge register	R	0x0000_0000	10.3.7.4/10-41
0x4_00B0	EOI—End of interrupt register	W	0x0000_0000	10.3.7.5/10-42
0x4_00C0–0x4_0FF0	Reserved	—	—	—
0x4_1000	FRR—Feature reporting register	R	0x0037_0002	10.3.1.1/10-16
0x4_1010	Reserved	—	—	—
0x4_1020	GCR—Global configuration register	R/W	0x0000_0000	10.3.1.2/10-17
0x4_1030	Reserved	—	—	—
0x4_1040–0x4_1070	Vendor reserved	—	—	—
0x4_1080	VIR—Vendor identification register	R	0x0000_0000	10.3.1.3/10-17
0x4_1090	PIR—Processor initialization register	R/W	0x0000_0000	10.3.1.4/10-18

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_10A0	IPIVPR0—IPI 0 vector/priority register	R/W	0x8000_0000	10.3.1.5/10-19
0x4_10B0	IPIVPR1—IPI 1 vector/priority register			
0x4_10C0	IPIVPR2—IPI 2 vector/priority register			
0x4_10D0	IPIVPR3—IPI 3 vector/priority register			
0x4_10E0	SVR—Spurious vector register	R/W	0x0000_FFFF	10.3.1.6/10-19
0x4_10F0	TFRR—Timer frequency reporting register	R/W	0x0000_0000	10.3.2.1/10-20
0x4_1100	GTCCR0—Global timer 0 current count register	R	0x0000_0000	10.3.2.2/10-21
0x4_1110	GTBCR0—Global timer 0 base count register	R/W	0x8000_0000	10.3.2.3/10-21
0x4_1120	GTVPR0—Global timer 0 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-22
0x4_1130	GTDR0—Global timer 0 destination register	R/W	0x0000_0001	10.3.2.5/10-23
0x4_1140	GTCCR1—Global timer 1 current count register	R	0x0000_0000	10.3.2.2/10-21
0x4_1150	GTBCR1—Global timer 1 base count register	R/W	0x8000_0000	10.3.2.3/10-21
0x4_1160	GTVPR1—Global timer 1 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-22
0x4_1170	GTDR1—Global timer 1 destination register	R/W	0x0000_0001	10.3.2.5/10-23
0x4_1180	GTCCR2—Global timer 2 current count register	R	0x0000_0000	10.3.2.2/10-21
0x4_1190	GTBCR2—Global timer 2 base count register	R/W	0x8000_0000	10.3.2.3/10-21
0x4_11A0	GTVPR2—Global timer 2 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-22
0x4_11B0	GTDR2—Global timer 2 destination register	R/W	0x0000_0001	10.3.2.5/10-23
0x4_11C0	GTCCR3—Global timer 3 current count register	R	0x0000_0000	10.3.2.2/10-21
0x4_11D0	GTBCR3—Global timer 3 base count register	R/W	0x8000_0000	10.3.2.3/10-21
0x4_11E0	GTVPR3—Global timer 3 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-22
0x4_11F0	GTDR3—Global timer 3 destination register	R/W	0x0000_0001	10.3.2.5/10-23
0x4_1200– 0x4_12F0	Reserved	—	—	—
0x4_1300	TCR—Timer control register	R/W	0x0000_0000	10.3.2.6/10-23
0x4_1310	IRQSR0—$\overline{\text{IRQ\_OUT}}$ summary register 0	R	0x0000_0000	10.3.3.1/10-26
0x4_1320	IRQSR1—$\overline{\text{IRQ\_OUT}}$ summary register 1	R	0x0000_0000	10.3.3.2/10-27
0x4_1330	CISR0—Critical Interrupt summary register 0	R	0x0000_0000	10.3.3.3/10-27
0x4_1340	CISR1—Critical Interrupt summary register 1	R	0x0000_0000	10.3.3.4/10-28
0x4_1350	PM0MR0—Performance monitor 0 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-29
0x4_1360	PM0MR1—Performance monitor 0 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-29
0x4_1370	PM1MR0—Performance monitor 1 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-29
0x4_1380	PM1MR1—Performance monitor 1 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-29

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1390	PM2MR0—Performance monitor 2 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-29
0x4_13A0	PM2MR1—Performance monitor 2 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-29
0x4_13B0	PM3MR0—Performance monitor 3 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-29
0x4_13C0	PM3MR1—Performance monitor 3 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-29
0x4_13D0–0x4_13F0	Reserved	—	—	—
0x4_1400	MSGR0—Message register 0	R/W	0x0000_0000	10.3.5.1/10-30
0x4_1410	MSGR1—Message register 1			
0x4_1420	MSGR2—Message register 2			
0x4_1430	MSGR3—Message register 3			
0x4_1440–0x4_14F0	Reserved	—	—	—
0x4_1500	MER—Message enable register	R/W	0x0000_0000	10.3.5.2/10-30
0x4_1510	MSR—Message status register	R/W	0x0000_0000	10.3.5.3/10-31
0x4_1520–0x4_FFF0	Reserved	—	—	—
PIC Register Address Map—Interrupt Source Configuration Registers				
0x5_0000	EIVPR0—External interrupt 0 (IRQ0) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0010	EIDR0—External interrupt 0 (IRQ0) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0020	EIVPR1—External interrupt 1 (IRQ1) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0030	EIDR1—External interrupt 1 (IRQ1) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0040	EIVPR2—External interrupt 2 (IRQ2) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0050	EIDR2—External interrupt 2 (IRQ2) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0060	EIVPR3—External interrupt 3 (IRQ3) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0070	EIDR3—External interrupt 3 (IRQ3) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0080	EIVPR4—External interrupt 4 (IRQ4) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0090	EIDR4—External interrupt 4 (IRQ4) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_00A0	EIVPR5—External interrupt 5 (IRQ5) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_00B0	EIDR5—External interrupt 5 (IRQ5) destination register	R/W	0x0000_0001	10.3.6.2/10-33

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_00C0	EIVPR6—External interrupt 6 (IRQ6) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_00D0	EIDR6—External interrupt 6 (IRQ6) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_00E0	EIVPR7—External interrupt 7 (IRQ7) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_00F0	EIDR7—External interrupt 7 (IRQ7) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0100	EIVPR8—External interrupt 8 (IRQ8) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0110	EIDR8—External interrupt 8 (IRQ8) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0120	EIVPR9—External interrupt 9 (IRQ9) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0130	EIDR9—External interrupt 9 (IRQ9) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0140	EIVPR10—External interrupt 10 (IRQ10) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0150	EIDR10—External interrupt 10 (IRQ10) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0160	EIVPR11—External interrupt 11 (IRQ11) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0170	EIDR11—External interrupt 11 (IRQ11) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0180–0x5_01F0	Reserved	—	—	—
0x5_0200	IIVPR0—Internal interrupt 0 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0210	IIDR0—Internal interrupt 0 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0220	IIVPR1—Internal interrupt 1 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0230	IIDR1—Internal interrupt 1 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0240	IIVPR2—Internal interrupt 2 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0250	IIDR2—Internal interrupt 2 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0260	IIVPR3—Internal interrupt 3 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0270	IIDR3—Internal interrupt 3 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0280	IIVPR4—Internal interrupt 4 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0290	IIDR4—Internal interrupt 4 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_02A0	IIVPR5—Internal interrupt 5 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_02B0	IIDR5—Internal interrupt 5 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_02C0	IIVPR6—Internal interrupt 6 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_02D0	IIDR6—Internal interrupt 6 destination register	R/W	0x0000_0001	10.3.6.4/10-35

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_02E0	IIVPR7—Internal interrupt 7 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_02F0	IIDR7—Internal interrupt 7 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0300	IIVPR8—Internal interrupt 8 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0310	IIDR8—Internal interrupt 8 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0320	IIVPR9—Internal interrupt 9 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0330	IIDR9—Internal interrupt 9 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0340	IIVPR10—Internal interrupt 10 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0350	IIDR10—Internal interrupt 10 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0360	IIVPR11—Internal interrupt 11 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0370	IIDR11—Internal interrupt 11 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0380	IIVPR12—Internal interrupt 12 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0390	IIDR12—Internal interrupt 12 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_03A0	IIVPR13—Internal interrupt 13 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_03B0	IIDR13—Internal interrupt 13 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_03C0	IIVPR14—Internal interrupt 14 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_03D0	IIDR14—Internal interrupt 14 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_03E0	IIVPR15—Internal interrupt 15 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_03F0	IIDR15—Internal interrupt 15 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0400	IIVPR16—Internal interrupt 16 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0410	IIDR16—Internal interrupt 16 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0420	IIVPR17—Internal interrupt 17 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0430	IIDR17—Internal interrupt 17 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0440	IIVPR18—Internal interrupt 18 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0450	IIDR18—Internal interrupt 18 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0460	IIVPR19—Internal interrupt 19 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0470	IIDR19—Internal interrupt 19 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0480	IIVPR20—Internal interrupt 20 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0490	IIDR20—Internal interrupt 20 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_04A0	IIVPR21—Internal interrupt 21 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_04B0	IIDR21—Internal interrupt 21 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_04C0	IIVPR22—Internal interrupt 22 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_04D0	IIDR22—Internal interrupt 22 destination register	R/W	0x0000_0001	10.3.6.4/10-35

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_04E0	IIVPR23—Internal interrupt 23 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_04F0	IIDR23—Internal interrupt 23 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0500	IIVPR24—Internal interrupt 24 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0510	IIDR24—Internal interrupt 24 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0520	IIVPR25—Internal interrupt 25 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0530	IIDR25—Internal interrupt 25 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0540	IIVPR26—Internal interrupt 26 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0550	IIDR126—Internal interrupt 26 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0560	IIVPR27—Internal interrupt 27 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0570	IIDR27—Internal interrupt 27 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0580	IIVPR28—Internal interrupt 28 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0590	IIDR28—Internal interrupt 28 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_05A0	IIVPR29—Internal interrupt 29 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_05B0	IIDR29—Internal interrupt 29 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_05C0	IIVPR30—Internal interrupt 30 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_05D0	IIDR30—Internal interrupt 30 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_05E0	IIVPR31—Internal interrupt 31 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_05F0	IIDR31—Internal interrupt 31 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0600– 0x5_15F0	Reserved	—	—	—
0x5_1600	MIVPR0—Messaging interrupt 0 (MSG 0) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-36
0x5_1610	MIDR0—Messaging interrupt 0 (MSG 0) destination register	R/W	0x0000_0001	10.3.6.6/10-37
0x5_1620	MIVPR1—Messaging interrupt 1 (MSG 1) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-36
0x5_1630	MIDR1—Messaging interrupt 1 (MSG 1) destination register	R/W	0x0000_0001	10.3.6.6/10-37
0x5_1640	MIVPR2—Messaging interrupt 2 (MSG 2) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-36
0x5_1650	MIDR2—Messaging interrupt 2 (MSG 2) destination register	R/W	0x0000_0001	10.3.6.6/10-37
0x5_1660	MIVPR3—Messaging interrupt 3 (MSG 3) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-36

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_1670	MIDR3—Messaging interrupt 3 (MSG 3) destination register	R/W	0x0000_0001	10.3.6.6/10-37
0x5_1680–0x5_FFF0	Reserved	—	—	—
PIC Register Address Map—Per-CPU Registers				
0x6_0000–0x6_0030	Reserved	—	—	—
0x6_0040	IPIDR0—P0 IPI 0 dispatch register	W	All zeros	10.3.7.1/10-39
0x6_0050	IPIDR1—P0 IPI 1 dispatch register			
0x6_0060	IPIDR2—P0 IPI 2 dispatch register			
0x6_0070	IPIDR3—P0 IPI 3 dispatch register			
0x6_0080	CTPR0—P0 current task priority register	R/W	0x0000_000F	10.3.7.2/10-40
0x6_0090	WHOAMI0—P0 who am I register	R	All zeros	10.3.7.3/10-41
0x6_00A0	IACK0—P0 interrupt acknowledge register	R	All zeros	10.3.7.4/10-41
0x6_00B0	EOI0—P0 end of interrupt register	W	All zeros	10.3.7.5/10-42
CPM Dual-Port RAM				
0x8_0000–0x8_3FFF	DPRAM1—Dual-port RAM	R/W	—	20.5/20-35
0x8_4000–0x8_7FFF	Reserved	—	—	—
0x8_8000–0x8_BFFF	DPRAM2—Dual-port RAM	R/W	—	20.5/20-35
0x8_C000–0x8_FFFF	Reserved	—	—	—
e500 Core Interface				
0x9_0000	CEAR—CPM Error Address Register	R	0x0000_0000	20.3.3.1.1/20-24
0x9_0004	CEER—CPM Error Event Register	R/W	0x0000	20.3.3.1.2/20-25
0x9_0006	CEMR—CPM Error Mask Register	R/W	0x0000	20.3.3.1.3/20-25
SDMA				
0x9_0050	SMAER—System bus address error register	R	0x0000_0000	26.1.1/26-2
0x9_0054	Reserved	—	—	—
0x9_0058	SMEVR—System bus event register	R/W	0x0000_0000	26.1.2/26-2
0x9_005C	SMCTR—System bus control register	R/W	0x3800_0000	26.1.3/26-3
0x9_0060	LMAER—Local bus address error register	R	0x0000_0000	26.1.1/26-2

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_0064	Reserved	—	—	—
0x9_0068	LMEVR—Local bus event register	R/W	0x0000_0000	26.1.2/26-2
0x9_006C	LMCTR—Local bus control register	R/W	0x3800_0000	26.1.3/26-3
Interrupt Controller				
0x9_0C00	SICR—CPM interrupt configuration register	R/W	0x0000_0000	21.5.1.1/21-9
0x9_0C02	Reserved	—	—	—
0x9_0C04	SIVEC—CPM interrupt vector register	R/W	0x0000_0000	21.5.1.5/21-14
0x9_0C08	SIPNR_H—CPM interrupt pending register (high)	R/W	0x0000_0000	21.5.1.3/21-11
0x9_0C0C	SIPNR_L—CPM interrupt pending register (low)	R/W	0x0000_0000	21.5.1.3/21-11
0x9_0C10	Reserved	—	—	—
0x9_0C14	SCPRR_H—CPM interrupt priority register (high)	R/W	0x0530_9770	21.5.1.2/21-10
0x9_0C18	SCPRR_L—CPM interrupt priority register (low)	R/W	0x0530_9770	21.5.1.2/21-10
0x9_0C1C	SIMR_H—CPM interrupt mask register (high)	R/W	0x0000_0000	21.5.1.4/21-13
0x9_0C20	SIMR_L—CPM interrupt mask register (low)	R/W	0x0000_0000	21.5.1.4/21-13
0x9_0C24	SIEXR—CPM external interrupt control register	R/W	0x0000_0000	21.5.1.6/21-15
0x9_0C28– 0x9_0C7F	Reserved	—	—	—
Clock				
0x9_0C80	SCCR—System clock control register	R/W	0x0000_0000	24.1/24-2
Input/Output Port				
0x9_0D00	PDIRA—Port A data direction register	R/W	0x0000_0000	45.2.3/45-3
0x9_0D04	PPARA—Port A pin assignment register	R/W	0x0000_0000	45.2.4/45-4
0x9_0D08	PSORA—Port A special options register	R/W	0x0000_0000	45.2.5/45-5
0x9_0D0C	PODRA—Port A open drain register	R/W	0x0000_0000	45.2.1/45-2
0x9_0D10	PDATA—Port A data register	R/W	0x0000_0000	45.2.2/45-2
0x9_0D14– 0x9_0D1F	Reserved	—	—	—
0x9_0D20	PDIRB—Port B data direction register	R/W	0x0000_0000	45.2.3/45-3
0x9_0D24	PPARB—Port B pin assignment register	R/W	0x0000_0000	45.2.4/45-4
0x9_0D28	PSORB—Port B special options register	R/W	0x0000_0000	45.2.5/45-5
0x9_0D2C	PODRB—Port B open drain register	R/W	0x0000_0000	45.2.1/45-2
0x9_0D30	PDATB—Port B data register	R/W	0x0000_0000	45.2.2/45-2

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_0D34–0x9_0D3F	Reserved	—	—	—
0x9_0D40	PDIRC—Port C data direction register	R/W	0x0000_0000	45.2.3/45-3
0x9_0D44	PPARC—Port C pin assignment register	R/W	0x0000_0000	45.2.4/45-4
0x9_0D48	PSORC—Port C special options register	R/W	0x0000_0000	45.2.5/45-5
0x9_0D4C	PODRC—Port C open drain register	R/W	0x0000_0000	45.2.1/45-2
0x9_0D50	PDATC—Port C data register	R/W	0x0000_0000	45.2.2/45-2
0x9_0D54–0x9_0D5F	Reserved	—	—	—
0x9_0D60	PDIRD—Port D data direction register	R/W	0x0000_0000	45.2.3/45-3
0x9_0D64	PPARD—Port D pin assignment register	R/W	0x0000_0000	45.2.4/45-4
0x9_0D68	PSORD—Port D special options register	R/W	0x0000_0000	45.2.5/45-5
0x9_0D6C	PODRD—Port D open drain register	R/W	0x0000_0000	45.2.1/45-2
0x9_0D70	PDATD—Port D data register	R/W	0x0000_0000	45.2.2/45-2
CPM Timers				
0x9_0D80	TGCR1—Timer 1 and timer 2 global configuration register	R/W	0x00	25.2.2/25-4
0x9_0D81	Reserved	—	—	—
0x9_0D84	TGCR2—Timer 3 and timer 4 global configuration register	R/W	0x00	25.2.2/25-4
0x9_0D85–0x9_0D8F	Reserved	—	—	—
0x9_0D90	TMR1—Timer 1 mode register	R/W	0x0000	25.2.3/25-6
0x9_0D92	TMR2—Timer 2 mode register	R/W	0x0000	25.2.3/25-6
0x9_0D94	TRR1—Timer 1 reference register	R/W	0x0000	25.2.4/25-7
0x9_0D96	TRR2—Timer 2 reference register	R/W	0x0000	25.2.4/25-7
0x9_0D98	TCR1—Timer 1 capture register	R/W	0x0000	25.2.5/25-8
0x9_0D9A	TCR2—Timer 2 capture register	R/W	0x0000	25.2.5/25-8
0x9_0D9C	TCN1—Timer 1 counter	R/W	0x0000	25.2.6/25-8
0x9_0D9E	TCN2—Timer 2 counter	R/W	0x0000	25.2.6/25-8
0x9_0DA0	TMR3—Timer 3 mode register	R/W	0x0000	25.2.3/25-6
0x9_0DA2	TMR4—Timer 4 mode register	R/W	0x0000	25.2.3/25-6
0x9_0DA4	TRR3—Timer 3 reference register	R/W	0x0000	25.2.4/25-7
0x9_0DA6	TRR4—Timer 4 reference register	R/W	0x0000	25.2.4/25-7

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_0DA8	TCR3—Timer 3 capture register	R/W	0x0000	25.2.5/25-8
0x9_0DAA	TCR4—Timer 4 capture register	R/W	0x0000	25.2.5/25-8
0x9_0DAC	TCN3—Timer 3 counter	R/W	0x0000	25.2.6/25-8
0x9_0DAE	TCN4—Timer 4 counter	R/W	0x0000	25.2.6/25-8
0x9_0DB0	TER1—Timer 1 event register	R/W	0x0000	25.2.7/25-8
0x9_0DB2	TER2—Timer 2 event register	R/W	0x0000	25.2.7/25-8
0x9_0DB4	TER3—Timer 3 event register	R/W	0x0000	25.2.7/25-8
0x9_0DB6	TER4—Timer 4 event register	R/W	0x0000	25.2.7/25-8
0x9_0DB8–0x9_101D	Reserved	—	—	—
FCC1				
0x9_1300	GFMR1—FCC1 general mode register	R/W	0x0000_0000	34.2/34-3
0x9_1304	FPSMR1—FCC1 protocol-specific mode register	R/W	0x0000_0000	(ATM) 35.13.2/35-89 (Ethernet) 40.18.1/40-21 (HDLC) 41.6/41-8
0x9_1308	FTODR1—FCC1 transmit on demand register	R/W	0x0000	34.6/34-9
0x9_130A	Reserved	—	—	—
0x9_130C	FDSR1—FCC1 data synchronization register	R/W	0x7E7E	34.5/34-8
0x9_130E	Reserved	—	—	—
0x9_1310	FCCE1—FCC1 event register	R/W	0x0000_0000	(ATM) 35.13.3/35-91 (Ethernet) 40.18.2/40-23 (HDLC) 41.9/41-14
0x9_1312	Reserved	—	—	—
0x9_1314	FCCM1—FCC1 mask register	R/W	0x0000_0000	(ATM) 35.13.3/35-91 (Ethernet) 40.18.2/40-23 (HDLC) 41.9/41-14
0x9_1316	Reserved	—	—	—
0x9_1318	FCCS1—FCC1 status register	R	0x00	41.10/41-16 (HDLC)
0x9_1319	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_131C	FTIRR1_PHY0—FCC1 transmit internal rate registers for PHY0	R/W	0x00	35.13.8/35-96 (ATM)
0x9_131D	FTIRR1_PHY1—FCC1 transmit internal rate registers for PHY1	R/W	0x00	35.13.8/35-96 (ATM)
0x9_131E	FTIRR1_PHY2—FCC1 transmit internal rate registers for PHY2	R/W	0x00	35.13.8/35-96 (ATM)
0x9_131F	FTIRR1_PHY3—FCC1 transmit internal rate registers for PHY3	R/W	0x00	35.13.8/35-96 (ATM)
FCC2				
0x9_1320	GFMR2—FCC2 general mode register	R/W	0x0000_0000	34.2/34-3
0x9_1324	FPSMR2—FCC2 protocol-specific mode register	R/W	0x0000_0000	(ATM) 35.13.2/35-89 (Ethernet) 40.18.1/40-21 (HDLC) 41.6/41-8
0x9_1328	FTODR2—FCC2 transmit on-demand register	R/W	0x0000	34.6/34-9
0x9_132A	Reserved	—	—	—
0x9_132C	FDSR2—FCC2 data synchronization register	R/W	0x7E7E	34.5/34-8
0x9_132E	Reserved	—	—	—
0x9_1330	FCCE2—FCC2 event register	R/W	0x0000_0000	(ATM) 35.13.3/35-91 (Ethernet) 40.18.2/40-23 (HDLC) 41.9/41-14
0x9_1332	Reserved	—	—	—
0x9_1334	FCCM2—FCC2 mask register	R/W	0x0000_0000	(ATM) 35.13.3/35-91 (Ethernet) 40.18.2/40-23 (HDLC) 41.9/41-14
0x9_1336	Reserved	—	—	—
0x9_1338	FCCS2—FCC2 status register	R	0x00	41.10/41-16 (HDLC)
0x9_1339	Reserved	—	0x00	—
0x9_133C	FTIRR2_PHY0—FCC2 transmit internal rate registers for PHY0	R/W	0x00	35.13.8/35-96 (ATM)
0x9_133D	FTIRR2_PHY1—FCC2 transmit internal rate registers for PHY1	R/W	0x00	35.13.8/35-96 (ATM)

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_133E	FTIRR2_PHY2—FCC2 transmit internal rate registers for PHY2	R/W	0x00	35.13.8/35-96 (ATM)
0x9_133F	FTIRR2_PHY3—FCC2 transmit internal rate registers for PHY3	R/W	0x00	35.13.8/35-96 (ATM)
FCC3				
0x9_1340	GFMR3—FCC3 general mode register	R/W	0x0000_0000	34.2/34-3
0x9_1344	FPSMR3—FCC3 protocol-specific mode register	R/W	0x0000_0000	(ATM) 35.13.2/35-89 (Ethernet) 40.18.1/40-21 (HDLC) 41.6/41-8
0x9_1348	FTODR3—FCC3 transmit on-demand register	R/W	0x0000	34.6/34-9
0x9_134A	Reserved	—	—	—
0x9_134C	FDSR3—FCC3 data synchronization register	R/W	0x7E7E	34.5/34-8
0x9_134E	Reserved	—	—	—
0x9_1350	FCCE3—FCC3 event register	R/W	0x0000_0000	(ATM) 35.13.3/35-91 (Ethernet) 40.18.2/40-23 (HDLC) 41.9/41-14
0x9_1352	Reserved	—	—	—
0x9_1354	FCCM3—FCC3 mask register	R/W	0x0000_0000	(ATM) 35.13.3/35-91 (Ethernet) 40.18.2/40-23 (HDLC) 41.9/41-14
0x9_1356	Reserved	—	—	—
0x9_1358	FCCS3—FCC3 status register	R	0x00	41.10/41-16 (HDLC)
0x9_1359–0x9_137F	Reserved	—	—	—
FCC1 (continued)				
0x9_1380	FIRPER1—FCC1 internal rate port enable register	R/W	0x0000_0000	35.13.5/35-92
0x9_1384	FIRER1—FCC1 internal rate event register	R/W	0x0000_0000	35.13.6/35-93
0x9_1388	FIRSR1_HI—FCC1 internal rate selection register:HI	R/W	0x0000_0000	35.13.7/35-94
0x9_138C	FIRSR1_LO— FCC1 internal rate selection register:LO	R/W	0x0000_0000	35.13.7/35-94

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_1390	GFEMR1—General FCC1 expansion mode register	R/W	0x00	34.3/34-7
0x9_1391–0x9_139F	Reserved	—	—	—
FCC2 (continued)				
0x9_13A0	FIRPER2—FCC2 internal rate port enable register	R/W	0x0000_0000	35.13.5/35-92
0x9_13A4	FIRER2—FCC2 internal rate event register	R/W	0x0000_0000	35.13.6/35-93
0x9_13A8	FIRSR2_HI—FCC2 internal rate selection register:HI	R/W	0x0000_0000	35.13.7/35-94
0x9_13AC	FIRSR2_LO—FCC2 internal rate selection register:LO	R/W	0x0000_0000	35.13.7/35-94
0x9_13B0	GFEMR2—General FCC2 expansion mode register	R/W	0x00	34.3/34-7
0x9_13B1–0x9_13CF	Reserved	—	—	—
FCC3 (continued)				
0x9_13D0	GFEMR3—General FCC3 expansion mode register	R/W	0x00	34.3/34-7
0x9_13D1–0x9_13FF	Reserved	—	—	—
TC Layer 1				
0x9_1400	TCMODE1—TC1 mode register			38.11.1.1/38-8
0x9_1402	CDSMR1—TC1 cell delineation state machine register			38.11.1.2/38-10
0x9_1404	TCER1—TC1 event register			38.11.1.3/38-10
0x9_1406	TC_RCC1—TC1 received cells counter			38.11.3.1/38-14
0x9_1408	TCMR1—TC1 mask register			38.11.1.4/38-11
0x9_140A	TC_FCC1—TC1 filtered cells counter			38.11.3.6/38-14
0x9_140C	TC_CCC1—TC1 corrected cells counter			38.11.3.4/38-14
0x9_140E	TC_ICC1—TC1 idle cells counter			38.11.3.5/38-14
0x9_1410	TC_TCC1—TC1 transmitted cells counter			38.11.3.2/38-14
0x9_1412	TC_ECC1—TC1 error cells counter			38.11.3.3/38-14
0x9_1414	Reserved	—	—	—
TC Layer 2				
0x9_1420	TCMODE2—TC2 mode register			38.11.1.1/38-8
0x9_1422	CDSMR2—TC2 cell delineation state machine register			38.11.1.2/38-10
0x9_1424	TCER2—TC2 event register			38.11.1.3/38-10
0x9_1426	TC_RCC2—TC2 received cells counter			38.11.3.1/38-14

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_1428	TCMR2—TC2 mask register			38.11.1.4/38-11
0x9_142A	TC_FCC2—TC2 filtered cells counter			38.11.3.6/38-14
0x9_142C	TC_CCC2—TC2 corrected cells counter			38.11.3.4/38-14
0x9_142E	TC_ICC2—TC2 idle cells counter			38.11.3.5/38-14
0x9_1430	TC_TCC2—TC2 transmitted cells counter			38.11.3.2/38-14
0x9_1432	TC_ECC2—TC2 error cells counter			38.11.3.3/38-14
0x9_1434	Reserved	—	—	—
TC Layer 3				
0x9_1440	TCMODE3—TC3 mode register			38.11.1.1/38-8
0x9_1442	CDSMR3—TC3 cell delineation state machine register			38.11.1.2/38-10
0x9_1444	TCER3—TC3 event register			38.11.1.3/38-10
0x9_1446	TC_RCC3—TC3 received cells counter			38.11.3.1/38-14
0x9_1448	TCMR3—TC3 mask register			38.11.1.4/38-11
0x9_144A	TC_FCC3—TC3 filtered cells counter			38.11.3.6/38-14
0x9_144C	TC_CCC3—TC3 corrected cells counter			38.11.3.4/38-14
0x9_144E	TC_ICC3—TC3 idle cells counter			38.11.3.5/38-14
0x9_1450	TC_TCC3—TC3 transmitted cells counter			38.11.3.2/38-14
0x9_1452	TC_ECC3—TC3 error cells counter			38.11.3.3/38-14
0x9_1454	Reserved	—	—	—
TC Layer 4				
0x9_1460	TCMODE4—TC4 mode register			38.11.1.1/38-8
0x9_1462	CDSMR4—TC4 cell delineation state machine register			38.11.1.2/38-10
0x9_1464	TCER4—TC4 event register			38.11.1.3/38-10
0x9_1466	TC_RCC4—TC4 received cells counter			38.11.3.1/38-14
0x9_1468	TCMR4—TC4 mask register			38.11.1.4/38-11
0x9_146A	TC_FCC4—TC4 filtered cells counter			38.11.3.6/38-14
0x9_146C	TC_CCC4—TC4 corrected cells counter			38.11.3.4/38-14
0x9_146E	TC_ICC4—TC4 idle cells counter			38.11.3.5/38-14
0x9_1470	TC_TCC4—TC4 transmitted cells counter			38.11.3.2/38-14
0x9_1472	TC_ECC4—TC4 error cells counter			38.11.3.3/38-14
0x9_1474	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
TC Layer 5				
0x9_1480	TCMODE5—TC5 mode register			38.11.1.1/38-8
0x9_1482	CDSMR5—TC5 cell delineation state machine register			38.11.1.2/38-10
0x9_1484	TCER5—TC5 event register			38.11.1.3/38-10
0x9_1486	TC_RCC5—TC5 received cells counter			38.11.3.1/38-14
0x9_1488	TCMR5—TC5 mask register			38.11.1.4/38-11
0x9_148A	TC_FCC5—TC5 filtered cells counter			38.11.3.6/38-14
0x9_148C	TC_CCC5—TC5 corrected cells counter			38.11.3.4/38-14
0x9_148E	TC_ICC5—TC5 idle cells counter			38.11.3.5/38-14
0x9_1490	TC_TCC5—TC5 transmitted cells counter			38.11.3.2/38-14
0x9_1492	TC_ECC5—TC5 error cells counter			38.11.3.3/38-14
0x9_1494	Reserved	—	—	—
TC Layer 6				
0x9_14A0	TCMODE6—TC6 mode register			38.11.1.1/38-8
0x9_14A2	CDSMR6—TC6 cell delineation state machine register			38.11.1.2/38-10
0x9_14A4	TCER6—TC6 event register			38.11.1.3/38-10
0x9_14A6	TC_RCC6—TC6 received cells counter			38.11.3.1/38-14
0x9_14A8	TCMR6—TC6 mask register			38.11.1.4/38-11
0x9_14AA	TC_FCC6—TC6 filtered cells counter			38.11.3.6/38-14
0x9_14AC	TC_CCC6—TC6 corrected cells counter			38.11.3.4/38-14
0x9_14AE	TC_ICC6—TC6 idle cells counter			38.11.3.5/38-14
0x9_14B0	TC_TCC6—TC6 transmitted cells counter			38.11.3.2/38-14
0x9_14B2	TC_ECC6—TC6 error cells counter			38.11.3.3/38-14
0x9_14B4	Reserved	—	—	—
TC Layer 7				
0x9_14C0	TCMODE7—TC7 mode register			38.11.1.1/38-8
0x9_14C2	CDSMR7—TC7 cell delineation state machine register			38.11.1.2/38-10
0x9_14C4	TCER7—TC7 event register			38.11.1.3/38-10
0x9_14C6	TC_RCC7—TC7 received cells counter			38.11.3.1/38-14
0x9_14C8	TCMR7—TC7 mask register			38.11.1.4/38-11
0x9_14CA	TC_FCC7—TC7 filtered cells counter			38.11.3.6/38-14
0x9_14CC	TC_CCC7—TC7 corrected cells counter			38.11.3.4/38-14

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_14CE	TC_ICC7—TC7 idle cells counter			38.11.3.5/38-14
0x9_14D0	TC_TCC7—TC7 transmitted cells counter			38.11.3.2/38-14
0x9_14D2	TC_ECC7—TC7 error cells counter			38.11.3.3/38-14
0x9_14D4	Reserved	—	—	—
TC Layer 8				
0x9_14E0	TCMODE8—TC8 mode register			38.11.1.1/38-8
0x9_14E2	CDSMR8—TC8 cell delineation state machine register			38.11.1.2/38-10
0x9_14E4	TCER8—TC8 event register			38.11.1.3/38-10
0x9_14E6	TC_RCC8—TC8 received cells counter			38.11.3.1/38-14
0x9_14E8	TCMR8—TC8 mask register			38.11.1.4/38-11
0x9_14EA	TC_FCC8—TC8 filtered cells counter			38.11.3.6/38-14
0x9_14EC	TC_CCC8—TC8 corrected cells counter			38.11.3.4/38-14
0x9_14EE	TC_ICC8—TC8 idle cells counter			38.11.3.5/38-14
0x9_14F0	TC_TCC8—TC8 transmitted cells counter			38.11.3.2/38-14
0x9_14F2	TC_ECC8—TC8 error cells counter			38.11.3.3/38-14
0x9_14F4	Reserved	—	—	—
TC Layer—General				
0x9_1500	TCGSR—TC general status register			38.11.2.2/38-13
0x9_1502	TCGER—TC general event register			38.11.2.1/38-12
BRGs 5–8				
0x9_15F0	BRGC5—BRG5 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_15F4	BRGC6—BRG6 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_15F8	BRGC7—BRG7 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_15FC	BRGC8—BRG8 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_1600– 0x9_185F	Reserved	—	—	—
I^2C				
0x9_1860	I2MOD—I^2C mode register	R/W	0x00	44.4.1/44-6
0x9_1861	Reserved	—	—	—
0x9_1864	I2ADD—I^2C address register	R/W	0x00	44.4.2/44-7
0x9_1865	Reserved	—	—	—
0x9_1868	II2BRG—I^2C BRG register	R/W	0x00	44.4.3/44-8

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_1869	Reserved	—	—	—
0x9_186C	I2COM—I^2C command register	R/W	0x00	44.4.5/44-9
0x9_186D	Reserved	—	—	—
0x9_1870	I2CER—I^2C event register	R/W	0x00	44.4.4/44-8
0x9_1871	Reserved	—	—	—
0x9_1874	II2CMR—^2C mask register	R/W	0x00	44.4.4/44-8
0x9_1875–0x9_19BF	Reserved	—	—	—
Communications Processor				
0x9_19C0	CPCR—Communications processor command register	R/W	0x0000_0000	20.4.1/20-30
0x9_19C4	RCCR—CP configuration register	R/W	0x0000_0000	20.3.6/20-28
0x9_19C8–0x9_19D5	Reserved	—	—	—
0x9_19D6	RTER—CP timers event register	R/W	0x0000	20.6.4/20-43
0x9_19DA	RTMR—CP timers mask register	R/W	0x0000	20.6.4/20-43
0x9_19DC	RTSCR—CP time-stamp timer control register	R/W	0x0000	20.3.7/20-29
0x9_19DE	Reserved	—	—	—
0x9_19E0	RTSR—CP time-stamp register	R/W	0x0000_0000	20.3.8/20-30
BRGs 1–4				
0x9_19F0	BRGC1—BRG1 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_19F4	BRGC2—BRG2 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_19F8	BRGC3—BRG3 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_19FC	BRGC4—BRG4 configuration register	R/W	0x0000_0000	24.2/24-3
SCC1				
0x9_1A00	GSMR_L1—SCC1 general mode register	R/W	0x0000_0000	27.2/27-3
0x9_1A04	GSMR_H1—SCC1 general mode register	R/W	0x0000_0000	27.2/27-3
0x9_1A08	PSMR1—SCC1 protocol-specific mode register	R/W	0x0000	27.2.1/27-9 28.16/28-13 (UART) 29.8/29-7 (HDLC) 30.11/30-10 (BISYNC) 31.9/31-9 (Transparent)
0x9_1A0A	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_1A0C	TODR1—SCC1 transmit-on-demand register	R/W	0x0000	27.2.3/27-10
0x9_1A0E	DSR1—SCC1 data synchronization register	R/W	0x7E7E	27.2.2/27-9
0x9_1A10	SCCE1—SCC1 event register	R/W		28.19/28-20 (UART) 29.11/29-13 (HDLC) 30.14/30-16 (BISYNC) 31.12/31-12 (Transparent)
0x9_1A14	SCCM1—SCC1 mask register	R/W	0x0000	
0x9_1A16	Reserved	—	—	—
0x9_1A17	SCCS1—SCC1 status register	R/W	0x00	28.20/28-22 (UART) 29.12/29-15 (HDLC) 30.15/30-16 (BISYNC) 31.13/31-13 (Transparent)
0x9_1A18–0x9_1A1F	Reserved	—	—	—
SCC2				
0x9_1A20	GSMR_L2—SCC2 general mode register (low)	R/W	0x0000_0000	27.2/27-3
0x9_1A24	GSMR_H2—SCC2 general mode register (high)	R/W	0x0000_0000	27.2/27-3
0x9_1A28	PSMR2—SCC2 protocol-specific mode register	R/W	0x0000	27.2.1/27-9 28.16/28-13 (UART) 29.8/29-7 (HDLC) 30.11/30-10 (BISYNC) 31.9/31-9 (Transparent)
0x9_1A2A	Reserved	—	—	—
0x9_1A2C	TODR2—SCC2 transmit-on-demand register	R/W	0x0000	27.2.3/27-10
0x9_1A2E	DSR2—SCC2 data synchronization register	R/W	0x7E7E	27.2.2/27-9
0x9_1A30	SCCE2—SCC2 event register	R/W		28.19/28-20 (UART) 29.11/29-13 (HDLC) 30.14/30-16 (BISYNC) 31.12/31-12 (Transparent)
0x9_1A34	SCCM2—SCC2 mask register	R/W	0x0000	

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_1A36	Reserved	—	—	—
0x9_1A37	SCCS2—SCC2 status register	R/W	0x00	28.20/28-22 (UART) 29.12/29-15 (HDLC) 30.15/30-16 (BISYNC) 31.13/31-13 (Transparent)
0x9_1A38–0x9_1A3F	Reserved	—	—	—
SCC3				
0x9_1A40	GSMR_L3—SCC3 general mode register	R/W	0x0000_0000	27.2/27-3
0x9_1A44	GSMR_H3—SCC3 general mode register	R/W	0x0000_0000	
0x9_1A48	PSMR3—SCC3 protocol-specific mode register	R/W	0x0000	27.2.1/27-9 28.16/28-13 (UART) 29.8/29-7 (HDLC) 30.11/30-10 (BISYNC) 31.9/31-9 (Transparent)
0x9_1A4A	Reserved	—	—	—
0x9_1A4C	TODR3—SCC3 transmit on demand register	R/W	0x0000	27.2.3/27-10
0x9_1A4E	DSR3—SCC3 data synchronization register	R/W	0x7E7E	27.2.2/27-9
0x9_1A50	SCCE3—SCC3 event register	R/W		28.19/28-20 (UART) 29.11/29-13 (HDLC) 30.14/30-16 (BISYNC) 31.12/31-12 (Transparent)
0x9_1A54	SCCM3—SCC3 mask register	R/W	0x0000	
0x9_1A56	Reserved	—	—	—
0x9_1A57	SCCS3—SCC3 status register	R/W	0x00	28.20/28-22 (UART) 29.12/29-15 (HDLC) 30.15/30-16 (BISYNC) 31.13/31-13 (Transparent)
0x9_1A58–0x9_1A5F	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
SCC4				
0x9_1A60	GSMR_L4—SCC4 general mode register	R/W	0x0000_0000	27.2/27-3
0x9_1A64	GSMR_H4—SCC4 general mode register	R/W	0x0000_0000	27.2/27-3
0x9_1A68	PSMR4—SCC4 protocol-specific mode register	R/W	0x0000	27.2.1/27-9 28.16/28-13 (UART) 29.8/29-7 (HDLC) 30.11/30-10 (BISYNC) 31.9/31-9 (Transparent)
0x9_1A6A	Reserved	—	—	—
0x9_1A6C	TODR4—SCC4 transmit on-demand register	R/W	0x0000	27.2.3/27-10
0x9_1A6E	DSR4—SCC4 data synchronization register	R/W	0x7E7E	27.2.2/27-9
0x9_1A70	SCCE4—SCC4 event register	R/W		28.19/28-20 (UART) 29.11/29-13 (HDLC) 30.14/30-16 (BISYNC) 31.12/31-12 (Transparent)
0x9_1A74	SCCM4—SCC4 mask register	R/W	0x0000	
0x9_1A76	Reserved	—	—	—
0x9_1A77	SCCS4—SCC4 status register	—	—	28.20/28-22 (UART) 29.12/29-15 (HDLC) 30.15/30-16 (BISYNC) 31.13/31-13 (Transparent)
0x9_1A78– 0x9_1A7F	Reserved	—	—	—
SPI				
0x9_1AA0	SPMODE—SPI mode register	R/W	0x0000	43.4.1/43-7
0x9_1AA2	Reserved	—	—	—
0x9_1AA6	SPIE—SPI event register	R/W	0x00	43.4.2/43-10
0x9_1AA7	Reserved	—	—	—
0x9_1AAA	SPIM—SPI mask register	R/W	0x00	43.4.2/43-10
0x9_1AAB	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_1AAD	SPCOM—SPI command register	W	0x00	43.4.3/43-11
0x9_1AA7–0x9_1AFF	Reserved	—	—	—
CPM Mux				
0x9_1B00	CMXSI1CR—CPM mux SI1 clock route register	R/W	0x00	23.4.2/23-12
0x9_1B02	CMXSI2CR—CPM mux SI2 clock route register	R/W	0x00	23.4.3/23-13
0x9_1B03	Reserved	—	—	—
0x9_1B04	CMXFCR—CPM mux FCC clock route register	R/W	0x0000_0000	23.4.4/23-14
0x9_1B08	CMXSCR—CPM mux SCC clock route register	R/W	0x0000_0000	23.4.5/23-16
0x9_1B0C	Reserved	—	—	—
0x9_1B0E	CMXUAR—CPM mux UTOPIA address register	R/W	0x0000	23.4.1/23-7
0x9_1B10–0x9_1B1F	Reserved	—	—	—
SI1 Registers				
0x9_1B20	SI1AMR—SI1 TDMA1 mode register	R/W	0x0000	22.5.2/22-18
0x9_1B22	SI1BMR—SI1 TDMB1 mode register	R/W	0x0000	
0x9_1B24	SI1CMR—SI1 TDMC1 mode register	R/W	0x0000	
0x9_1B26	SI1DMR—SI1 TDMD1 mode register	R/W	0x0000	
0x9_1B28	SI1GMR—SI1 global mode register	R/W	0x00	22.5.1/22-18
0x9_1B29	Reserved	—	—	—
0x9_1B2A	SI1CMDR—SI1 command register	R/W	0x00	22.5.4/22-25
0x9_1B2B	Reserved	—	—	—
0x9_1B2C	SI1STR—SI1 status register	R/W	0x00	22.5.5/22-26
0x9_1B2D	Reserved	—	—	—
0x9_1B2E	SI1RSR—SI1 RAM shadow address register	R/W	0x0000	22.5.3/22-24
MCC1 Registers				
0x9_1B30	MCCE1—MCC1 event register	R/W	0x0000	33.12.1/33-30
0x9_1B32	Reserved	—	—	—
0x9_1B34	MCCM1—MCC1 mask register	R/W	0x0000	33.12.1/33-30
0x9_1B36	Reserved	—	—	—
0x9_1B38	MCCF1—MCC1 configuration register	R/W	0x00	33.10/33-27
0x9_1B39–0x9_1B3F	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
SI2 Registers				
0x9_1B40	SI2AMR—SI2 TDMA2 mode register	R/W	0x0000	22.5.2/22-18
0x9_1B42	SI2BMR—SI2 TDMB2 mode register	R/W	0x0000	22.5.2/22-18
0x9_1B44	SI2CMR—SI2 TDMC2 mode register	R/W	0x0000	22.5.2/22-18
0x9_1B46	SI2DMR—SI2 TDMD2 mode register	R/W	0x0000	22.5.2/22-18
0x9_1B48	SI2GMR—SI2 global mode register	R/W	0x00	22.5.1/22-18
0x9_1B49	Reserved	—	—	—
0x9_1B4A	SI2CMDR—SI2 command register	R/W	0x00	22.5.4/22-25
0x9_1B4B	Reserved	—	—	—
0x9_1B4C	SI2STR—SI2 status register	R/W	0x00	22.5.5/22-26
0x9_1B4D	Reserved	—	—	—
0x9_1B4E	SI2RSR—SI2 RAM shadow address register	R/W	0x0000	22.5.3/22-24
MCC2 Registers				
0x9_1B50	MCCE2—MCC2 event register	R/W	0x0000	33.12.1/33-30
0x9_1B52	Reserved	—	—	—
0x9_1B54	MCCM2—MCC2 mask register	R/W	0x0000	33.12.1/33-30
0x9_1B56	Reserved	—	—	—
0x9_1B58	MCCF2—MCC2 configuration register	R/W	0x00	33.10/33-27
0x9_1B59– 0x9_1FFF	Reserved	—	—	—
SI1 RAM				
0x9_2000– 0x9_21FF	SI1TxRAM—SI 1 transmit routing RAM			22.4.3/22-10
0x9_2200– 0x9_23FF	Reserved	—	—	—
0x9_2400– 0x9_25FF	SI1RxRAM—SI 1 receive routing RAM			22.4.3/22-10
0x9_2600– 0x9_27FF	Reserved	—	—	—
SI2 RAM				
0x9_2800– 0x9_29FF	SI2TxRAM—SI 2 transmit routing RAM			22.4.3/22-10
0x9_2A00– 0x9_2BFF	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x9_2C00–0x9_2DFF	SI2RxRAM—SI 2 receive routing RAM			22.4.3/22-10
0x9_2E00–0x9_3FFF	Reserved	—	—	—
Instruction RAM				
0xA_0000–0xA_7FFF	Dual-port RAM (instruction RAM only)			
0xA_8000–0xB_FFFF	Reserved	—	—	—
RapidIO Registers				
RapidIO Architectural Registers				
0xC_0000	DIDCAR—Device identity capability register	R	0x0003_0002	16.3.1.1/16-13
0xC_0004	DICAR—Device information capability register	R	0x8070_0020	16.3.1.2/16-14
0xC_0008	AIDCAR—Assembly identity capability register	R/W	0x0000_0000	16.3.1.3/16-14
0xC_000C	AICAR—Assembly information capability register	R/W	0x0000_0100	16.3.1.4/16-15
0xC_0010	PEFCAR—Processing element features capability register	R	0xE088_0009	16.3.1.5/16-15
0xC_0014	SPICAR—Switch port information capability register	R	0x0000_0100	16.3.1.6/16-17
0xC_0018	SOCAR—Source operations capability register	R	0x0600_FCF0	16.3.1.7/16-17
0xC_001C	DOCAR—Destination operations capability register	R	0x0600_FCF4	16.3.1.8/16-19
0xC_0040	MSR—Mailbox command and status register	R	0x0000_0000	16.3.1.9/16-22
0xC_0044	PWDCSR—Port-write and doorbell command and status register	R	0x0000_0020	16.3.1.10/16-22
0xC_004C	PELLCCSR—Processing element logical layer control command and status register	R	0x0000_0001	16.3.1.11/16-24
0xC_0058	Reserved	R	0x0000_0000	—
0xC_005C	LCSBA1CSR—Local configuration space base address 1 command and status register	R/W	0x0000_0000	16.3.1.12/16-24
0xC_0060	BDIDCSR—Base device ID command and status register	R/W	0x00[cfg]_0000	16.3.1.13/16-25
0xC_0068	HBDIDLCSR—Host base device ID lock command and status register	Special	0x0000_FFFF	16.3.1.14/16-26
0xC_006C	CTCSR—Component tag command and status register	R/W	0x0000_0000	16.3.1.15/16-26
0xC_0100	PMBH0CSR—8/16 LP-LVDS port maintenance block header 0 command and status register	R	0x0000_0002	16.3.1.16/16-27
0xC_0120	PLTOCCSR—Port link time-out control command and status register	R/W	0xFFFF_FF00	16.3.1.17/16-27

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xC_0124	PRTOCCSR—Port response time-out control command and status register	R/W	0xFFFF_FF00	16.3.1.18/16-28
0xC_013C	PGCCSR—Port general control command and status register	R/W	0x[cfg]000_0000	16.3.1.19/16-29
0xC_0140	PLMREQCSR—Port link maintenance request command and status register	R/W	0x0000_0000	16.3.1.20/16-29
0xC_0144	PLMRESPCSR—Port link maintenance response command and status register	R	0x0000_0000	16.3.1.21/16-30
0xC_0148	PLASCSR—Port local ackID status command and status register	R/W	0x0000_0000	16.3.1.22/16-31
0xC_0158	PESCSR—Port error and status command and status register	R/W	0x0000_0000	16.3.1.23/16-32
0xC_015C	PCCSR—Port control command and status register	R/W	0x4400_0000	16.3.1.24/16-33
Implementation Registers				
0xD_0000	CR—Configuration register	R/W	0x0000_0001	16.3.2.1.1/16-34
0xD_0010	PCR—Port configuration register	R/W	0x0000_0010	16.3.2.1.2/16-35
0xD_0014	PEIR—Port error injection register	R/W	0x0000_0000	16.3.2.1.3/16-35
ATMU Registers				
0xD_0C00	ROWTAR0—RapidIO outbound window translation address register 0	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0C10	ROWAR0—RapidIO outbound window attributes register 0	R/W	0x8004_401F	16.3.2.2.3/16-39
0xD_0C20	ROWTAR1—RapidIO outbound window translation address register 1	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0C28	ROWBAR1—RapidIO outbound window base address register 1	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0C30	ROWAR1—RapidIO outbound window attributes register 1	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0C40	ROWTAR2—RapidIO outbound window translation address register 2	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0C48	ROWBAR2—RapidIO outbound window base address register 2	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0C50	ROWAR2—RapidIO outbound window attributes register 2	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0C60	ROWTAR3—RapidIO outbound window translation address register 3	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0C68	ROWBAR3—RapidIO outbound window base address register 3	R/W	0x0000_0000	16.3.2.2.2/16-38

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_0C70	ROWAR3—RapidIO outbound window attributes register 3	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0C80	ROWTAR4—RapidIO outbound window translation address register 4	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0C88	ROWBAR4—RapidIO outbound window base address register 4	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0C90	ROWAR4—RapidIO outbound window attributes register 4	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0CA0	ROWTAR5—RapidIO outbound window translation address register 5	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0CA8	ROWBAR5—RapidIO outbound window base address register 5	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0CB0	ROWAR5—RapidIO outbound window attributes register 5	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0CC0	ROWTAR6—RapidIO outbound window translation address register 6	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0CC8	ROWBAR6—RapidIO outbound window base address register 6	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0CD0	ROWAR6—RapidIO outbound window attributes register 6	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0CE0	ROWTAR7—RapidIO outbound window translation address register 7	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0CE8	ROWBAR7—RapidIO outbound window base address register 7	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0CF0	ROWAR7—RapidIO outbound window attributes register 7	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0D00	ROWTAR8—RapidIO outbound window translation address register 8	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0D08	ROWBAR8—RapidIO outbound window base address register 8	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0D10	ROWAR8—RapidIO outbound window attributes register 8	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0D60	RIWTAR4—RapidIO inbound window translation address register 4	R/W	0x0000_0000	16.3.2.2.4/16-41
0xD_0D68	RIWBAR4—RapidIO inbound window base address register 4	R/W	0x0000_0000	16.3.2.2.5/16-41
0xD_0D70	RIWAR4—RapidIO inbound window attributes register 4	R/W	0x0004_401F	16.3.2.2.6/16-42
0xD_0D80	RIWTAR3—RapidIO inbound window translation address register 3	R/W	0x0000_0000	16.3.2.2.4/16-41

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_0D88	RIWBAR3—RapidIO inbound window base address register 3	R/W	0x0000_0000	16.3.2.2.5/16-41
0xD_0D90	RIWAR3—RapidIO inbound window attributes register 3	R/W	0x0004_401F	16.3.2.2.6/16-42
0xD_0DA0	RIWTAR2—RapidIO inbound window translation address register 2	R/W	0x0000_0000	16.3.2.2.4/16-41
0xD_0DA8	RIWBAR2—RapidIO inbound window base address register 2	R/W	0x0000_0000	16.3.2.2.5/16-41
0xD_0DB0	RIWAR2—RapidIO inbound window attributes register 2	R/W	0x0004_401F	16.3.2.2.6/16-42
0xD_0DC0	RIWTAR1—RapidIO inbound window translation address register 1	R/W	0x0000_0000	16.3.2.2.4/16-41
0xD_0DC8	RIWBAR1—RapidIO inbound window base address register 1	R/W	0x0000_0000	16.3.2.2.5/16-41
0xD_0DD0	RIWAR1—RapidIO inbound window attributes register 1	R/W	0x0004_401F	16.3.2.2.6/16-42
0xD_0DE0	RIWTAR0—RapidIO inbound window translation address register 0	R/W	0x0000_0000	16.3.2.2.4/16-41
0xD_0DF0	RIWAR0—RapidIO inbound window attributes register 0	R/W	0x8004_401F	16.3.2.2.6/16-42
Error Management Registers				
0xD_0E00	PNFEDR—Port notification/fatal error detect register	R/W	0x0000_0000	16.3.2.3.1/16-44
0xD_0E04	PNFEDiR—Port notification/fatal error detect disable register	R/W	0x0000_0000	16.3.2.3.2/16-47
0xD_0E08	PNFEIER—Port notification/fatal error interrupt enable register	R/W	0x0000_0000	16.3.2.3.3/16-49
0xD_0E0C	PECSR—Port error capture status register	R/W	0x0000_0000	16.3.2.3.4/16-52
0xD_0E10	EPCR0—Error packet capture register 0	R/W	0x0000_0000	16.3.2.3.5/16-52
0xD_0E14	EPCR1—Error packet capture register 1	R/W	0x0000_0000	16.3.2.3.6/16-53
0xD_0E18	EPCR2—Error packet capture register 2	R/W	0x0000_0000	16.3.2.3.16/16-58
0xD_0E20	PREDR—Port recoverable error detect register	R/W	0x0000_0000	16.3.2.3.23/16-61
0xD_0E28	PERTR—Port error recovery threshold register	R/W	0x00FF_0000	16.3.2.3.24/16-64
0xD_0E2C	PRTR—Port retry threshold register	R/W	0x00FF_0000	16.3.2.3.25/16-64
RapidIO Message Unit				
RapidIO Outbound Message Registers				
0xD_1000	OMR—Outbound mode register	R/W	0x0000_0000	16.3.3.1.1/16-65
0xD_1004	OSR—Outbound status register	R/W	0x0000_0000	16.3.3.1.2/16-67
0xD_100C	ODQDPAR—Outbound descriptor queue dequeue pointer address register	R/W	0x0000_0000	16.3.3.1.3/16-68
0xD_1014	OSAR—Outbound source address register	R/W	0x0000_0000	16.3.3.1.4/16-69

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_1018	ODPR—Outbound destination port register	R/W	0x0000_0000	16.3.3.1.5/16-70
0xD_101C	ODATR—Outbound destination attributes register	R/W	0x0006_0000	16.3.3.1.6/16-70
0xD_1020	ODCR—Outbound double-word count register	R/W	0x0000_0000	16.3.3.1.7/16-71
0xD_1028	ODQEPAR—Outbound descriptor queue enqueue pointer address register	R/W	0x0000_0000	16.3.3.1.8/16-72
RapidIO Inbound Message Registers				
0xD_1060	IMR—Inbound mailbox mode register	R/W	0x0000_0000	16.3.3.2.1/16-72
0xD_1064	ISR—Inbound mailbox status register	R/W	0x0000_0000	16.3.3.2.2/16-74
0xD_106C	IFQDPAR—Inbound frame queue dequeue pointer address register	R/W	0x0000_0000	16.3.3.2.3/16-75
0xD_1074	IFQEPAR—Inbound frame queue enqueue pointer address register	R/W	0x0000_0000	16.3.3.2.4/16-76
RapidIO Doorbell Registers				
0xD_1460	DMR—Doorbell mode register	R/W	0x0000_0000	16.3.3.3.1/16-77
0xD_1464	DSR—Doorbell status register	R/W	0x0000_0000	16.3.3.3.2/16-78
0xD_146C	DQDPAR—Doorbell queue dequeue pointer address register	R/W	0x0000_0000	16.3.3.3.3/16-79
0xD_1474	DQEPAR—Doorbell queue enqueue pointer address register	R/W	0x0000_0000	16.3.3.3.4/16-80
RapidIO Port-Write Registers				
0xD_14E0	PWMR—Port-write mode register	R/W	0x0000_0000	16.3.3.4.1/16-81
0xD_14E4	PWSR—Port-write status register	R/W	0x0000_0000	16.3.3.4.2/16-82
0xD_14EC	PWQBAR—Port-write queue base address register	R/W	0x0000_0000	16.3.3.4.3/16-83
Global Utilities Registers				
Power-On Reset Configuration Values				
0xE_0000	PORPLLSR—POR PLL ratio status register	R	0x00*nn*_00*nn*	17.4.1.1/17-4
0xE_0004	PORBMSR—POR boot mode status register	R	0x*nnnn*_0000	17.4.1.2/17-5
0xE_0008	PORIMPSCR—POR I/O impedance status and control register	R/W	0x000*n*_007F	17.4.1.3/17-6
0xE_000C	PORDEVSR—POR I/O device status register	R	See ref.	17.4.1.4/17-7
0xE_0010	PORDBGMSR—POR debug mode status register	R	See ref.	17.4.1.5/17-9
0xE_0020	GPPORCR—General-purpose POR configuration register	R	See ref.	17.4.1.6/17-9

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
Signal Multiplexing and GPIO Controls				
0xE_0030	GPIOCR—GPIO control register	R/W	0x0000_0000	17.4.1.7/17-10
0xE_0040	GPOUTDR—General-purpose output data register	R/W	0x0000_0000	17.4.1.8/17-11
0xE_0050	GPINDR—General-purpose input data register	R	0x*nnnn*_0000	17.4.1.9/17-12
0xE_0060	PMUXCR—Alternate function signal multiplex control	R/W	0x0000_0000	17.4.1.9/17-12
Device Disables				
0xE_0070	DEVDISR—Device disable control	R/W	0x0000_0000	17.4.1.11/17-13
Power Management Registers				
0xE_0080	POWMGTCSR—Power management status and control register	R/W	0x0000_0000	17.4.1.12/17-15
Interrupt Reporting				
0xE_0090	MCPSUMR—Machine check summary register	Read/ Clear	0x0000_0000	17.4.1.13/17-16
Version Registers				
0xE_00A0	PVR—Processor version register	R	e500 processor version	17.4.1.14/17-17
0xE_00A4	SVR—System version register	R	MPC8560 system version	17.4.1.15/17-18
Debug Control				
0xE_0E00	CLKOCR—Clock out select register	R/W	0x0000_0000	17.4.1.16/17-18
0xE_0E10	DDRDLLCR—DDR DLL control register	R/W	0x0000_0000	17.4.1.17/17-19
0xE_0E20	LBDLLCR—LBC DLL control register	R/W	0x0000_0000	17.4.1.18/17-20
Performance Monitor Control Registers				
0xE_1000	PMGC0—Performance monitor global control register	R/W	0x0000_0000	18.3.2.1/18-5
0xE_1010	PMLCA0—Performance monitor local control register A0	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1014	PMLCB0—Performance monitor local control register B0	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1018	PMC0 (upper)—Performance monitor counter 0 upper	R/W	0x0000_0000	18.3.3.1/18-9
0xE_101C	PMC0 (lower)—Performance monitor counter 0 lower	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1020	PMLCA1—Performance monitor local control register A1	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1024	PMLCB1—Performance monitor local control register B1	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1028	PMC1—Performance monitor counter 1	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1030	PMLCA2—Performance monitor local control register A2	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1034	PMLCB2—Performance monitor local control register B2	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1038	PMC2—Performance monitor counter 2	R/W	0x0000_0000	18.3.3.1/18-9

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xE_1040	PMLCA3—Performance monitor local control register A3	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1044	PMLCB3—Performance monitor local control register B3	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1048	PMC3—Performance monitor counter 3	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1050	PMLCA4—Performance monitor local control register A4	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1054	PMLCB4—Performance monitor local control register B4	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1058	PMC4—Performance monitor counter 4	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1060	PMLCA5—Performance monitor local control register A5	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1064	PMLCB5—Performance monitor local control register B5	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1068	PMC5—Performance monitor counter 5	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1070	PMLCA6—Performance monitor local control register A6	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1074	PMLCB6—Performance monitor local control register B6	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1078	PMC6—Performance monitor counter 6	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1080	PMLCA7—Performance monitor local control register A7	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1084	PMLCB7—Performance monitor local control register B7	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1088	PMC7—Performance monitor counter 7	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1090	PMLCA8—Performance monitor local control register A8	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1094	PMLCB8—Performance monitor local control register B8	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1098	PMC8—Performance monitor counter 8	R/W	0x0000_0000	18.3.3.1/18-9
Debug and Watchpoint Monitor Registers				
Watchpoint Monitor Registers				
0xE_2000	WMCR0—Watchpoint monitor control register 0	R/W	0x0000_0000	19.3.1.1/19-11
0xE_2004	WMCR1—Watchpoint monitor control register 1	R/W	0x0000_0000	19.3.1.1/19-11
0xE_200C	WMAR—Watchpoint monitor address register	R/W	0x0000_0000	19.3.1.2/19-13
0xE_2014	WMAMR—Watchpoint monitor address mask register	R/W	0x0000_0000	19.3.1.3/19-14
0xE_2018	WMTMR—Watchpoint monitor transaction mask register	R/W	0x0000_0000	19.3.1.4/19-14
0xE_201C	WMSR—Watchpoint monitor status register	R/W	0x0000_0000	19.3.1.5/19-16
Trace Buffer Registers				
0xE_2040	TBCR0—Trace buffer control register	R/W	0x0000_0000	19.3.2.1/19-16
0xE_2044	TBCR1—Trace buffer control register	R/W	0x0000_0000	19.3.2.1/19-16
0xE_204C	TBAR—Trace buffer address register	R/W	0x0000_0000	19.3.2.2/19-19
0xE_2054	TBAMR—Trace buffer address mask register	R/W	0x0000_0000	19.3.2.3/19-19
0xE_2058	TBTMR—Trace buffer transaction mask register	R/W	0x0000_0000	19.3.2.4/19-20

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xE_205C	TBSR—Trace buffer status register	R/W	0x0000_0000	19.3.2.5/19-21
0xE_2060	TBACR—Trace buffer access control register	R/W	0x0000_0000	19.3.2.6/19-22
0xE_2064	TBADHR—Trace buffer access data high register	R/W	0x0000_0000	19.3.2.7/19-22
0xE_2068	TBADR—Trace buffer access data register	R/W	0x0000_0000	19.3.2.8/19-23
Context ID Registers				
0xE_20A0	PCIDR—Programmed context ID register	R/W	0x0000_0000	19.3.3.1/19-24
0xE_20A4	CCIDR—Current context ID register	R/W	0x0000_0000	19.3.3.2/19-24
Other Registers				
0xE_20B0	TOSR—Trigger output source register	R/W	0x0000_0000	19.3.4.1/19-25

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

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

Chapter 3
Signal Descriptions

This chapter describes the MPC8560 external signals. It is organized into the following sections:

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

NOTE

A bar over a signal name indicates that the signal is active low, such as $\overline{\text{IRQ\_OUT}}$ (interrupt out). Active-low signals are referred to as asserted (active) when they are low and negated when they are high. Signals that are not active low, such as IRQ (interrupt input), are referred to as asserted when they are high and negated when they are low.

Internal signals throughout this document are shown 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.

3.1 Signals Overview

The 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 Integrated Processor Hardware Specifications* for a pinout diagram showing pin numbers and a listing of all the electrical and mechanical specifications.

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

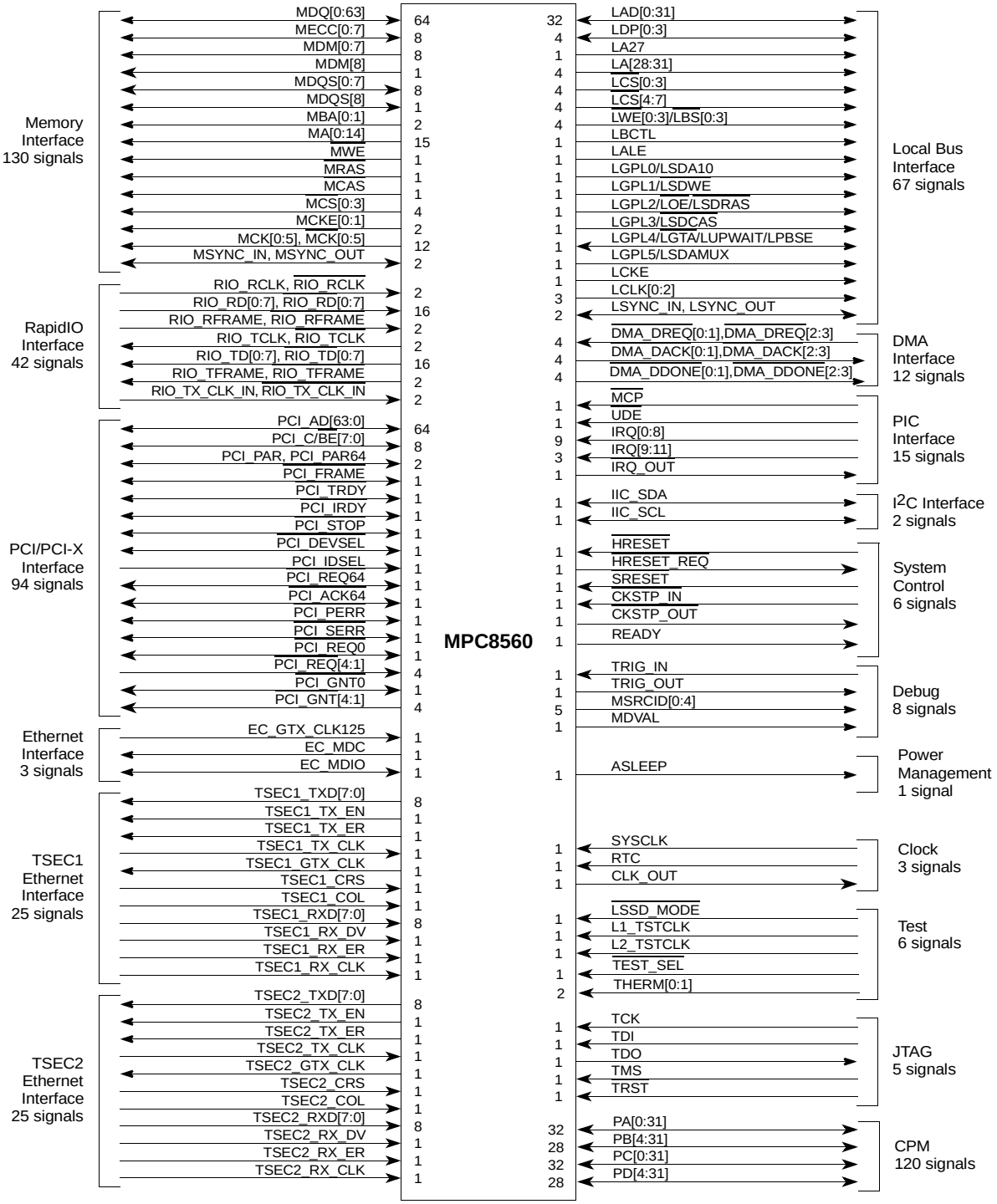


Figure 3-1. MPC8560 Signal Groupings

The following tables provide summaries of signal functions. Table 3-1 provides a summary of the signals grouped by function, and Table 3-2 provides the summary list of the signals grouped alphabetically. These tables detail the signal name, interface, alternate functions, number of

signals, and whether the signal is an input, output, or bidirectional. The direction of the multiplexed signals applies for the primary signal function listed in the left-most column of the table for that row (and does not apply for the state of the reset configuration signals). Finally, the table provides a pointer to the table where the signal function is described.

Table 3-1. MPC8560 Signal Reference by Functional Block

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
MDQ[0:63]	DDR data	DDR memory	—	64	I/O	9-3/9-5
MECC[0:7]	DDR error correcting code	DDR memory	—	8	I/O	9-3/9-5
MDM[0:7]	DDR data mask	DDR memory	—	8	O	9-3/9-5
MDM8	DDR ECC data mask	DDR memory	—	1	O	9-3/9-5
MDQS[0:7]	DDR data strobe	DDR memory	—	8	I/O	9-3/9-5
MDQS8	DDR ECC data strobe	DDR memory	—	1	I/O	9-3/9-5
MBA[0:1]	DDR bank select	DDR memory	—	2	O	9-3/9-5
MA[0:14]	DDR address	DDR memory	—	15	O	9-3/9-5
$\overline{\text{MWE}}$	DDR write enable	DDR memory	—	1	O	9-3/9-5
$\overline{\text{MRAS}}$	DDR row address strobe	DDR memory	—	1	O	9-3/9-5
$\overline{\text{MCAS}}$	DDR column address strobe	DDR memory	—	1	O	9-3/9-5
$\overline{\text{MCS}}$[0:3]	DDR chip select (2/DIMM)	DDR memory	—	4	O	9-3/9-5
MCKE[0:1]	DDR clock enable	DDR memory	—	2	O	9-4/9-8
MCK[0:5], $\overline{\text{MCK}}$[0:5]	DDR differential clocks (3 pairs/DIMM)	DDR memory	—	12	O	9-4/9-8
MSYNC_IN, MSYNC_OUT	DDR DLL synchronization in/out	DDR memory	—	2	I/O	9-4/9-8
RIO_RCLK, $\overline{\text{RIO\_RCLK}}$	RapidIO receive clocks	RapidIO	—	2	I	16-5/16-8
RIO_RD[0:7], $\overline{\text{RIO\_RD}}$[0:7]	RapidIO receive data	RapidIO	—	16	I	16-5/16-8
RIO_RFRAME, $\overline{\text{RIO\_RFRAME}}$	RapidIO receive frame	RapidIO	—	2	I	16-5/16-8
RIO_TCLK, $\overline{\text{RIO\_TCLK}}$	RapidIO transmit clock	RapidIO	—	2	O	16-5/16-8
RIO_TD[0:7], $\overline{\text{RIO\_TD}}$[0:7]	RapidIO transmit data	RapidIO	—	16	O	16-5/16-8
RIO_TFRAME, $\overline{\text{RIO\_TFRAME}}$	RapidIO transmit frame	RapidIO	—	2	O	16-5/16-8

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
RIO_TX_CLK_IN RIO_TX_CLK_IN	RapidIO transmit clock in	RapidIO	—	2	I	16-5/16-8
PCI_AD[63:0]	PCI address/data	PCI/X	—	64	I/O	15-2/15-8
PCI_C/BE[7:0]	PCI command/byte enable	PCI/X	—	8	I/O	15-2/15-8
PCI_PAR	PCI parity	PCI/X	—	1	I/O	15-2/15-8
PCI_PAR64	PCI parity 64	PCI/X	—	1	I/O	15-2/15-8
PCI_FRAME	PCI frame	PCI/X	—	1	I/O	15-2/15-8
PCI_TRDY	PCI target ready	PCI/X	—	1	I/O	15-2/15-8
PCI_IRDY	PCI initiator ready	PCI/X	—	1	I/O	15-2/15-8
PCI_STOP	PCI stop	PCI/X	—	1	I/O	15-2/15-8
PCI_DEVSEL	PCI device select	PCI/X	—	1	I/O	15-2/15-8
PCI_IDSEL	PCI initial device select	PCI/X	—	1	I	15-2/15-8
PCI_REQ64	PCI request 64	PCI/X	cfg_pci_width	1	I/O	15-2/15-8
PCI_ACK64	PCI acknowledge 64	PCI/X	—	1	I/O	15-2/15-8
PCI_PERR	PCI parity error	PCI/X	—	1	I/O	15-2/15-8
PCI_SERR	PCI system error	PCI/X	—	1	I/O	15-2/15-8
PCI_REQ0	PCI request 0	PCI/X	—	1	I/O	15-2/15-8
PCI_REQ[4:1]	PCI request 4–1	PCI/X	—	4	I	15-2/15-8
PCI_GNT0	PCI grant 0	PCI/X	—	1	I/O	15-2/15-8
PCI_GNT[4:1]	PCI grant 4–1	PCI/X	cfg_pci_mode cfg_pci_debug cfg_pci_arbiter cfg_pci_impd	4	O	15-2/15-8
EC_GTX_CLK125	Gigabit reference clock	Gigabit clock	—	1	I	13-3/13-14
EC_MDC	Ethernet management data clock	Ethernet management	cfg_tsec_reduce	1	O	13-3/13-14
EC_MDIO	Ethernet management data in/out	Ethernet management	—	1	I/O	13-3/13-14
TSEC1_TXD7	TSEC1 transmit data 7	TSEC1	cfg_tsec1	1	O	13-3/13-14
TSEC1_TXD[6:4]	TSEC1 transmit data 6–4	TSEC1	cfg_rom_loc[0:2]	3	O	13-3/13-14
TSEC1_TXD3	TSEC1 transmit data 3	TSEC1	—	1	O	13-3/13-14
TSEC1_TXD2	TSEC1 transmit data 2	TSEC1	—	1	O	13-3/13-14
TSEC1_TXD1	TSEC1 transmit data 1	TSEC1	—	1	O	13-3/13-14

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
TSEC1_TXD0	TSEC1 transmit data 0	TSEC1	—	1	O	13-3/13-14
TSEC1_TX_EN	TSEC1 transmit enable	TSEC1	—	1	O	13-3/13-14
TSEC1_TX_ER	TSEC1 transmit error	TSEC1	—	1	O	13-3/13-14
TSEC1_TX_CLK	TSEC1 transmit clock in	TSEC1	—	1	I	13-3/13-14
TSEC1_GTX_CLK	TSEC1 transmit clock out	TSEC1	_	1	O	13-3/13-14
TSEC1_CRS	TSEC1 carrier sense	TSEC1	—	1	I	13-3/13-14
TSEC1_COL	TSEC1 collision detect	TSEC1	—	1	I	13-3/13-14
TSEC1_RXD[7:0]	TSEC1 receive data	TSEC1	—	8	I	13-3/13-14
TSEC1_RX_DV	TSEC1 receive data valid	TSEC1	—	1	I	13-3/13-14
TSEC1_RX_ER	TSEC1 receiver error	TSEC1	—	1	I	13-3/13-14
TSEC1_RX_CLK	TSEC1 receive clock	TSEC1	—	1	I	13-3/13-14
TSEC2_TXD7	TSEC2 transmit data 7	TSEC2	cfg_tsec2	1	O	13-3/13-14
TSEC2_TXD[6:5]	TSEC2 transmit data 6–5	TSEC2	cfg_lb_hold[0:1]	2	O	13-3/13-14
TSEC2_TXD[4:2]	TSEC2 transmit data 4–2	TSEC2	cfg_dev_ID[7:5]	3	O	13-3/13-14
TSEC2_TXD1	TSEC2 transmit data 1	TSEC2	—	1	O	13-3/13-14
TSEC2_TXD0	TSEC2 transmit data 0	TSEC2	—	1	O	13-3/13-14
TSEC2_TX_EN	TSEC2 transmit enable	TSEC2	—	1	O	13-3/13-14
TSEC2_TX_ER	TSEC2 transmit error	TSEC2	—	1	O	13-3/13-14
TSEC2_TX_CLK	TSEC2 transmit clock in	TSEC2	—	1	I	13-3/13-14
TSEC2_GTX_CLK	TSEC2 transmit clock out	TSEC2	—	1	O	13-3/13-14
TSEC2_CRS	TSEC2 carrier sense	TSEC2	—	1	I	13-3/13-14
TSEC2_COL	TSEC2 collision detect	TSEC2	—	1	I	13-3/13-14
TSEC2_RXD[0:7]	TSEC2 receive data	TSEC2	—	8	I	13-3/13-14
TSEC2_RX_DV	TSEC2 receive data valid	TSEC2	—	1	I	13-3/13-14
TSEC2_RX_ER	TSEC2 receiver error	TSEC2	—	1	I	13-3/13-14
TSEC2_RX_CLK	TSEC2 receive clock	TSEC2	—	1	I	13-3/13-14
LAD[0:31]	LBC address/data	LBC	cfg_gpporcr	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

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
LCS[0:4]	LBC chip select 0–4	LBC	—	5	O	12-1/12-5
LCS5	LBC chip select 5	LBC	DMA_DREQ2	1	O	12-1/12-5
LCS6	LBC chip select 6	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	cfg_pci_hold[0:1]	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_clk1	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

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{DMA\_DDONE2}}$	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
$\overline{\text{MCP}}$	Machine check processor	PIC	—	1	I	10-6/10-8
$\overline{\text{UDE}}$	Unconditional debug event	PIC	—	1	I	10-6/10-8
IRQ[0:8]	External interrupt 0–8	PIC	—	9	I	10-6/10-8
IRQ9	External interrupt 9	PIC	$\overline{\text{DMA\_DREQ3}}$	1	I	10-6/10-8
IRQ10	External interrupt 10	PIC	$\overline{\text{DMA\_DACK3}}$	1	I	10-6/10-8
IRQ11	External interrupt 11	PIC	$\overline{\text{DMA\_DDONE3}}$	1	I	10-6/10-8
$\overline{\text{IRQ\_OUT}}$	Interrupt output	PIC	—	1	O	10-6/10-8
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	4-2/4-2
$\overline{\text{HRESET\_REQ}}$	Hard reset request	System control	—	1	O	4-2/4-2
$\overline{\text{SRESET}}$	Soft reset	System control	—	1	I	4-2/4-2
$\overline{\text{CKSTP\_IN}}$	Checkstop in	System control	—	1	I	17.2/17-1
$\overline{\text{CKSTP\_OUT}}$	Checkstop out	System control	—	1	O	17.2/17-1
READY	Device ready	System control	TRIG_OUT	1	O	4-2/4-2
TRIG_IN	Watchpoint trigger in	Debug	—	1	I	19-2/19-6
TRIG_OUT	Watchpoint trigger out	Debug	READY	1	O	19-2/19-6
MSRCID[0:1]	Memory debug source port ID 0–1	Debug	cfg_mem_debug cfg_ddr_debug	2	O	9-6/9-10
MSRCID[2:4]	Memory debug source port ID 2–4	Debug	—	3	O	19-2/19-6
MDVAL	Memory debug data valid	Debug	—	1	O	19-2/19-6
ASLEEP	Asleep	Power mgmt	—	1	O	17.2/17-1
SYSCLK	System clock/PCI clock	Clock	—	1	I	4-3/4-3
RTC	Real time clock	Clock	—	1	I	4-3/4-3
CLK_OUT	Clock out	Clock	—	1	O	17.2/17-1
$\overline{\text{LSSD\_MODE}}$	LSSD mode	Test	—	1	I	19-2/19-6
L1_TSTCLK	L1 test clock	Test	—	1	I	19-2/19-6
L2_TSTCLK	L2 test clock	Test	—	1	I	19-2/19-6
$\overline{\text{TEST\_SEL}}$	Test select	Test	—	1	I	19-2/19-6

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
THERM[0:1]	Thermal resistor access	Test	—	2	I	19-2/19-6
TCK	Test clock	JTAG	—	1	I	19-2/19-6
TDI	Test data in	JTAG	—	1	I	19-2/19-6
TDO	Test data out	JTAG	—	1	O	19-2/19-6
TMS	Test mode select	JTAG	—	1	I	19-2/19-6
$\overline{\text{TRST}}$	Test reset	JTAG	—	1	I	19-2/19-6
PA[0:31]	Port A	CPM	See Chapter 45, “Parallel I/O Ports”	32	I/O	45-5/45-8
PB[4:31]	Port B	CPM	See Chapter 45, “Parallel I/O Ports”	28	I/O	45-6/45-12
PC[0:31]	Port C	CPM	See Chapter 45, “Parallel I/O Ports”	32	I/O	45-7/45-15
PD[4:31]	Port D	CPM	See Chapter 45, “Parallel I/O Ports”	28	I/O	45-8/45-17

Table 3-2. MPC8560 Alphabetical Signal Reference

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
ASLEEP	Asleep	Power mgmt	—	1	O	17.2/17-1
$\overline{\text{CKSTP\_IN}}$	Checkstop in	System control	—	1	I	17.2/17-1
$\overline{\text{CKSTP\_OUT}}$	Checkstop out	System control	—	1	O	17.2/17-1
CLK_OUT	Clock out	Clock	—	1	O	17.2/17-1
$\overline{\text{DMA\_DACK2}}$	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	13-3/13-14
EC_MDC	Ethernet management data clock	Ethernet management	cfg_tsec_reduce	1	O	13-3/13-14

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
EC_MDIO	Ethernet management data in/out	Ethernet management	—	1	I/O	13-3/13-14
$\overline{\text{HRESET}}$	Hard reset	System control	—	1	I	4-2/4-2
$\overline{\text{HRESET\_REQ}}$	Hard reset request	System control	—	1	O	4-2/4-2
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	10-6/10-8
IRQ10	External interrupt 10	PIC	$\overline{\text{DMA\_DACK3}}$	1	I	10-6/10-8
IRQ11	External interrupt 11	PIC	$\overline{\text{DMA\_DDONE3}}$	1	I	10-6/10-8
IRQ9	External interrupt 9	PIC	$\overline{\text{DMA\_DREQ3}}$	1	I	10-6/10-8
$\overline{\text{IRQ\_OUT}}$	Interrupt output	PIC	—	1	O	10-6/10-8
L1_TSTCLK	L1 test clock	Test	—	1	I	19-2/19-6
L2_TSTCLK	L2 test clock	Test	—	1	I	19-2/19-6
LA27	LBC burst address	LBC	cfg_cpu_boot	1	O	12-1/12-5
LAD[0:31]	LBC address/data	LBC	cfg_gpporcr	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
$\overline{\text{LCS5}}$	LBC chip select 5	LBC	$\overline{\text{DMA\_DREQ2}}$	1	O	12-1/12-5
$\overline{\text{LCS6}}$	LBC chip select 6	LBC	$\overline{\text{DMA\_DACK2}}$	1	O	12-1/12-5
$\overline{\text{LCS7}}$	LBC chip select 7	LBC	$\overline{\text{DMA\_DDONE2}}$	1	O	12-1/12-5
$\overline{\text{LCS}}$[0:4]	LBC Chip select 0–4	LBC	—	5	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/$\overline{\text{LSDWE}}$	LBC GP line 1/SDRAM write enable	LBC	cfg_rio_clk1	1	O	12-1/12-5
LGPL2/$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$	LBC GP line 2/output enable/SDRAM RAS	LBC	cfg_core_pll1	1	O	12-1/12-5

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
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	19-2/19-6
LSYNC_IN, LSYNC_OUT	LBC DLL synchronization	LBC	—	2	I/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	cfg_pci_hold[0:1]	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
MA[0:14]	DDR address	DDR memory	—	15	O	9-3/9-5
MBA[0:1]	DDR bank select	DDR memory	—	2	O	9-3/9-5
MCAS	DDR column address strobe	DDR memory	—	1	O	9-3/9-5
MCKE[0:1]	DDR clock enable	DDR memory	—	2	O	9-4/9-8
MCK[0:5], MCK[0:5]	DDR differential clocks (3 pairs/DIMM)	DDR memory	—	12	O	9-4/9-8
MCP	Machine check processor	PIC	—	1	I	10-6/10-8
MCS[0:3]	DDR chip select (2/DIMM)	DDR memory	—	4	O	9-3/9-5
MDM8	DDR ECC data mask	DDR memory	—	1	O	9-3/9-5
MDM[0:7]	DDR data mask	DDR memory	—	8	O	9-3/9-5
MDQS8	DDR ECC data strobe	DDR memory	—	1	I/O	9-3/9-5
MDQS[0:7]	DDR data strobe	DDR memory	—	8	I/O	9-3/9-5
MDQ[0:63]	DDR data	DDR memory	—	64	I/O	9-3/9-5
MDVAL	Memory debug data valid	Debug	—	1	O	19-2/19-6
MECC[0:7]	DDR error correcting code	DDR memory	—	8	I/O	9-3/9-5
MRAS	DDR row address strobe	DDR memory	—	1	O	9-3/9-5
MSRCID[0:1]	Memory debug source port ID 0–1	Debug	cfg_mem_debug cfg_ddr_debug	2	O	9-6/9-10

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
MSRCID[2:4]	Memory debug source port ID 2–4	Debug	—	3	O	19-2/19-6
MSYNC_IN, MSYNC_OUT	DDR DLL synchronization in/out	DDR memory	—	2	I/O	9-4/9-8
$\overline{\text{MWE}}$	DDR write enable	DDR memory	—	1	O	9-3/9-5
PA[0:31]	Port A	CPM	See Chapter 45, “Parallel I/O Ports”	32	I/O	45-5/45-8
PB[4:31]	Port B	CPM	See Chapter 45, “Parallel I/O Ports”	28	I/O	45-6/45-12
$\overline{\text{PCI\_ACK64}}$	PCI acknowledge 64	PCI/X	—	1	I/O	15-2/15-8
PCI_AD[63:0]	PCI address/data	PCI/X	—	64	I/O	15-2/15-8
PCI_C/$\overline{\text{BE}}$[7:0]	PCI command/byte enable	PCI/X	—	8	I/O	15-2/15-8
$\overline{\text{PCI\_DEVSEL}}$	PCI device select	PCI/X	—	1	I/O	15-2/15-8
$\overline{\text{PCI\_FRAME}}$	PCI frame	PCI/X	—	1	I/O	15-2/15-8
$\overline{\text{PCI\_GNT0}}$	PCI grant 0	PCI/X	—	1	I/O	15-2/15-8
$\overline{\text{PCI\_GNT}}$[4:1]	PCI grant 4–1	PCI/X	cfg_pci_mode cfg_pci_debug cfg_pci_arbiter cfg_pci_impd	4	O	15-2/15-8
PCI_IDSEL	PCI initial device select	PCI/X	—	1	I	15-2/15-8
$\overline{\text{PCI\_IRDY}}$	PCI initiator ready	PCI/X	—	1	I/O	15-2/15-8
PCI_PAR	PCI parity	PCI/X	—	1	I/O	15-2/15-8
PCI_PAR64	PCI parity 64	PCI/X	—	1	I/O	15-2/15-8
$\overline{\text{PCI\_PERR}}$	PCI parity error	PCI/X	—	1	I/O	15-2/15-8
$\overline{\text{PCI\_REQ0}}$	PCI request 0	PCI/X	—	1	I/O	15-2/15-8
$\overline{\text{PCI\_REQ64}}$	PCI request 64	PCI/X	cfg_pci_width	1	I/O	15-2/15-8
$\overline{\text{PCI\_REQ}}$[4:1]	PCI request 4–1	PCI/X	—	4	I	15-2/15-8
$\overline{\text{PCI\_SERR}}$	PCI system error	PCI/X	—	1	I/O	15-2/15-8
$\overline{\text{PCI\_STOP}}$	PCI stop	PCI/X	—	1	I/O	15-2/15-8
$\overline{\text{PCI\_TRDY}}$	PCI target ready	PCI/X	—	1	I/O	15-2/15-8
PC[0:31]	Port C	CPM	See Chapter 45, “Parallel I/O Ports”	32	I/O	45-7/45-15
PD[4:31]	Port D	CPM	See Chapter 45, “Parallel I/O Ports”	28	I/O	45-8/45-17
READY	Device ready	System control	TRIG_OUT	1	O	4-2/4-2

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
RIO_RCLK, $\overline{\text{RIO\_RCLK}}$	RapidIO receive clocks	RapidIO	—	2	I	16-5/16-8
RIO_RD[0:7], $\overline{\text{RIO\_RD}}$[0:7]	RapidIO receive data	RapidIO	—	16	I	16-5/16-8
RIO_RFRAME, $\overline{\text{RIO\_RFRAME}}$	RapidIO receive frame	RapidIO	—	2	I	16-5/16-8
RIO_TCLK, $\overline{\text{RIO\_TCLK}}$	RapidIO transmit clock	RapidIO	—	2	O	16-5/16-8
RIO_TD[0:7], $\overline{\text{RIO\_TD}}$[0:7]	RapidIO transmit data	RapidIO	—	16	O	16-5/16-8
RIO_TFRAME, $\overline{\text{RIO\_TFRAME}}$	RapidIO transmit frame	RapidIO	—	2	O	16-5/16-8
RIO_TX_CLK_IN $\overline{\text{RIO\_TX\_CLK\_IN}}$	RapidIO transmit clock in	RapidIO	—	2	I	16-5/16-8
RTC	Real time clock	Clock	—	1	I	4-3/4-3
$\overline{\text{SRESET}}$	Soft reset	System control	—	1	I	4-2/4-2
SYSCLK	System clock/PCI clock	Clock	—	1	I	4-3/4-3
TCK	Test clock	JTAG	—	1	I	19-2/19-6
TDI	Test data in	JTAG	—	1	I	19-2/19-6
TDO	Test data out	JTAG	—	1	O	19-2/19-6
$\overline{\text{TEST\_SEL}}$	Test select	Test	—	1	I	19-2/19-6
THERM[0:1]	Thermal resistor access	Test	—	2	I	19-2/19-6
TMS	Test mode select	JTAG	—	1	I	19-2/19-6
TRIG_IN	Watchpoint trigger in	Debug	—	1	I	19-2/19-6
TRIG_OUT	Watchpoint trigger out	Debug	READY	1	O	19-2/19-6
$\overline{\text{TRST}}$	Test reset	JTAG	—	1	I	19-2/19-6
TSEC1_COL	TSEC1 collision detect	TSEC1	—	1	I	13-3/13-14
TSEC1_CRS	TSEC1 carrier sense	TSEC1	—	1	I	13-3/13-14
TSEC1_GTX_CLK	TSEC1 transmit clock out	TSEC1	—	1	O	13-3/13-14
TSEC1_RXD[7:0]	TSEC1 receive data	TSEC1	—	8	I	13-3/13-14
TSEC1_RX_CLK	TSEC1 receive clock	TSEC1	—	1	I	13-3/13-14
TSEC1_RX_DV	TSEC1 receive data valid	TSEC1	—	1	I	13-3/13-14
TSEC1_RX_ER	TSEC1 receiver error	TSEC1	—	1	I	13-3/13-14
TSEC1_TXD0	TSEC1 transmit data 0	TSEC1	—	1	O	13-3/13-14

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
TSEC1_TXD1	TSEC1 transmit data 1	TSEC1	—	1	O	13-3/13-14
TSEC1_TXD2	TSEC1 transmit data 2	TSEC1	—	1	O	13-3/13-14
TSEC1_TXD3	TSEC1 transmit data 3	TSEC1	—	1	O	13-3/13-14
TSEC1_TXD[6:4]	TSEC1 transmit data 6–4	TSEC1	cfg_rom_loc[0:2]	3	O	13-3/13-14
TSEC1_TXD7	TSEC1 transmit data 7	TSEC1	cfg_tsec1	1	O	13-3/13-14
TSEC1_TX_CLK	TSEC1 transmit clock in	TSEC1	—	1	I	13-3/13-14
TSEC1_TX_EN	TSEC1 transmit enable	TSEC1	—	1	O	13-3/13-14
TSEC1_TX_ER	TSEC1 transmit error	TSEC1	—	1	O	13-3/13-14
TSEC2_COL	TSEC2 collision detect	TSEC2	—	1	I	13-3/13-14
TSEC2_CRS	TSEC2 carrier sense	TSEC2	—	1	I	13-3/13-14
TSEC2_GTX_CLK	TSEC2 transmit clock out	TSEC2	—	1	O	13-3/13-14
TSEC2_RXD[0:7]	TSEC2 receive data	TSEC2	—	8	I	13-3/13-14
TSEC2_RX_CLK	TSEC2 receive clock	TSEC2	—	1	I	13-3/13-14
TSEC2_RX_DV	TSEC2 receive data valid	TSEC2	—	1	I	13-3/13-14
TSEC2_RX_ER	TSEC2 receiver error	TSEC2	—	1	I	13-3/13-14
TSEC2_TXD0	TSEC2 transmit data 0	TSEC2	—	1	O	13-3/13-14
TSEC2_TXD1	TSEC2 transmit data 1	TSEC2	—	1	O	13-3/13-14
TSEC2_TXD[4:2]	TSEC2 transmit data 4–2	TSEC2	cfg_dev_ID[7:5]	3	O	13-3/13-14
TSEC2_TXD[6:5]	TSEC2 transmit data 6–5	TSEC2	cfg_lb_hold[0:1]	2	O	13-3/13-14
TSEC2_TXD7	TSEC2 transmit data 7	TSEC2	cfg_tsec2	1	O	13-3/13-14
TSEC2_TX_CLK	TSEC2 transmit clock in	TSEC2	—	1	I	13-3/13-14
TSEC2_TX_EN	TSEC2 transmit enable	TSEC2	—	1	O	13-3/13-14
TSEC2_TX_ER	TSEC2 transmit error	TSEC2	—	1	O	13-3/13-14
$\overline{\text{UDE}}$	Unconditional debug event	PIC	—	1	I	10-6/10-8

3.2 Configuration Signals Sampled at Reset

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

Note that throughout this document, the reset configuration signals are described as being sampled at the negation of $\overline{\text{HRESET}}$. However, there is a setup and hold time for these signals relative to

the rising edge of $\overline{\text{HRESET}}$, as described in the *MPC8560 Integrated Processor Hardware Specifications*. Note that the PLL configuration signals have different setup and hold time requirements than the other reset configuration signals.

The reset configuration signals are multiplexed with other functional signals. The values on these signals during reset are interpreted to be logic one or zero, regardless of whether the functional signal name is defined as active-low. Most of the reset configuration signals have internal pull-up resistors so that if the signals are not driven, the default value is high (a one), as shown in the table. Some signals do not have pull-up resistors and must be driven high or low during the reset period. For details about all the signals that require external pull-up resistors, see the *MPC8560 Integrated Processor Hardware Specifications*.

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 45, "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
PCI	$\overline{\text{PCI\_REQ64}}$	cfg_pci_width	1
	$\overline{\text{PCI\_GNT4}}$	cfg_pci_mode	1
	$\overline{\text{PCI\_GNT3}}$	cfg_pci_debug	1
	$\overline{\text{PCI\_GNT2}}$	cfg_pci_arbiter	1
	$\overline{\text{PCI\_GNT1}}$	cfg_pci_impd	1
Ethernet Management	EC_MDC	cfg_tsec_reduce	1
TSEC1	TSEC1_TXD7	cfg_tsec1	1
	TSEC1_TXD[6:4]	cfg_rom_loc[0:2]	111
TSEC2	TSEC2_TXD7	cfg_tsec2	1
	TSEC2_TXD[6:5]	cfg_lb_hold[0:1]	11
	TSEC2_TXD[4:2]	cfg_dev_ID[7:5]	111

Table 3-3. MPC8560 Reset Configuration Signals (continued)

Functional Interface	Functional Signal Name	Reset Configuration Name	Default
LBC	LA27	cfg_cpu_boot	1
	LA[28:31]	cfg_sys_pll[0:3]	Must be driven
	$\overline{\text{LWE}}$[0:1]/$\overline{\text{LBS}}$[0:1]	cfg_pci_hold[0:1]	11
	$\overline{\text{LWE}}$[2:3]/$\overline{\text{LBS}}$[2:3]	cfg_host_agt0 cfg_host_agt1	11
	LALE	cfg_core_pll0	Must be driven
	LGPL0/LSDA10	cfg_rio_clk0	1
	LGPL1/$\overline{\text{LSDWE}}$	cfg_rio_clk1	1
	LGPL2/$\overline{\text{LOE}}$/$\overline{\text{LSDRAS}}$	cfg_core_pll1	Must be driven
	LGPL3/$\overline{\text{LSDCAS}}$	cfg_boot_seq0	1
	LGPL5/LSDAMUX	cfg_boot_seq1	1
	LAD[0:31]	cfg_gpporcr	Indeterminate if not driven (no default)
Debug	MSRCID0	cfg_mem_debug	1
	MSRCID1	cfg_ddr_debug	1

3.3 Output Signal States During Reset

When a system reset is recognized ($\overline{\text{HRESET}}$ 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
DDR Memory	MDM[0:8]	High-Z
DDR Memory	MBA[0:1]	High-Z
DDR Memory	MA[0:14]	High-Z
DDR Memory	$\overline{\text{MWE}}$	High-Z
DDR Memory	$\overline{\text{MRAS}}$	High-Z

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

Interface	Signal	State During Reset
DDR Memory	$\overline{\text{MCAS}}$	High-Z
DDR Memory	$\overline{\text{MCS}}$[0:3]	High-Z
DDR Memory	MCKE[0:1]	Driven Low
DDR Memory	MCK[0:5], $\overline{\text{MCK}}$[0:5]	High-Z
RapidIO	RIO_TCLK, RIO_TCLK	Driven (toggling)
RapidIO	RIO_TD[0:7], $\overline{\text{RIO\_TD}}$[0:7]	High-Z
RapidIO	RIO_TFRAME, $\overline{\text{RIO\_TFRAME}}$	High-Z
TSEC1	TSEC1_TXD[3:0]	Input—reset config (test only)
TSEC1	TSEC1_TX_EN	Driven Low
TSEC1	TSEC1_TX_ER	High-Z
TSEC1	TSEC1_GTX_CLK	High-Z
TSEC2	TSEC2_TXD[1:0]	High-Z
TSEC2	TSEC2_TX_EN	Driven Low
TSEC2	TSEC2_TX_ER	High-Z
TSEC2	TSEC2_GTX_CLK	High-Z
LBC	LCLK[0:2]	High-Z
LBC	$\overline{\text{LCS}}$[0:5]	High-Z
LBC	$\overline{\text{DMA\_DACK2}}$/$\overline{\text{LCS6}}$ $\overline{\text{DMA\_DDONE2}}$/LCS7	High-Z
LBC	LBCTL	Input—reset config (test only)
DMA	$\overline{\text{DMA\_DACK}}$[0:1]	High-Z
DMA	$\overline{\text{DMA\_DACK2}}$/$\overline{\text{LCS6}}$ $\overline{\text{DMA\_DDONE2}}$/LCS7	High-Z
DMA	$\overline{\text{DMA\_DDONE0}}$	High-Z
DMA	$\overline{\text{DMA\_DDONE1}}$	Driven (test only)
PIC	IRQ8	Driven (test only)
PIC	$\overline{\text{IRQ\_OUT}}$	High-Z
System Control	$\overline{\text{HRESET\_REQ}}$	High-Z
System Control	$\overline{\text{CKSTP\_OUT}}$	High-Z
Debug	TRIG_OUT/READY	Input—reset config (test only)
Debug	MSRCID[2:4]	High-Z
Debug	MDVAL	High-Z

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

Interface	Signal	State During Reset
Power Mgmt	ASLEEP	Input—reset config (test only)
Clock	CLK_OUT	High-Z

Chapter 4
Reset, Clocking, and Initialization

This chapter describes the reset, clocking, and some overall initialization of the MPC8560, including a definition of the reset configurations signals and the options they select. Additionally, the configuration, control, and status registers are described. Note that chapters in this book describe specific initialization aspects for individual blocks.

4.1 Overview

The reset, clocking, and control signals provide many options for the operation of the MPC8560. Additionally, many modes are selected with reset configuration signals during the assertion of a hard reset (assertion of $\overline{\text{HRESET}}$).

4.2 External Signal Descriptions

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

Table 4-1. Signal Summary

Signal	I/O	Description	References (Section/Page)
$\overline{\text{HRESET}}$	I	Hard reset input. Causes a power-on reset (POR) sequence	4.4.1.2/4-9
$\overline{\text{HRESET\_REQ}}$	O	Hard reset request output. An internal block requests that $\overline{\text{HRESET}}$ be asserted	—
$\overline{\text{SRESET}}$	I	Soft reset input. Causes *mcp* assertion to the core and a soft reset to the CPM	4.4.1.1/4-9
READY	O	The MPC8560 has completed the reset operation and is not in a power-down (nap, doze or sleep) or debug state.	4.4.2/4-9
SYSCLK	I	Primary clock input to the MPC8560	4.4.4.1/4-23
RTC	I	Real time clock input	4.4.4.4/4-24

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

4.2.1 System Control Signals

Table 4-2 describes some of the system control signals of the MPC8560. Section 4.4.3, "Power-On Reset Configuration," describes the signals that also function as reset configuration signals. Note that the CKSTP_IN and CKSTP_OUT signals are described in Chapter 17, "Global Utilities."

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

Signal	I/O	Description	
HRESET	I	Hard reset. Causes the MPC8560 to abort all current internal and external transactions and set all registers to their default values. HRESET may be asserted completely asynchronously with respect to all other signals.	
		State Meaning	Asserted/Negated—See Chapter 3, "Signal Descriptions," and Section 4.4.3, "Power-On Reset Configuration," for more information on the interpretation of the other MPC8560 signals during reset.
		Timing	Assertion/Negation—The *MPC8560 Integrated Processor Hardware Specifications* gives specific timing information for this signal and the reset configuration signals.
HRESET_REQ	O	Hard reset request. Indicates to the board (system in which the MPC8560 is embedded) 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 (see Section 11.4.5, "Boot Sequencer Mode") has triggered a request for hard reset. Negated—Indicates no reset request
		Timing	Assertion/Negation—May occur anytime, synchronous to the core complex bus clock. Once asserted, HRESET_REQ does not negate until HRESET is asserted.
SRESET	I	Soft reset. 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 causes a core checkstop. SRESET need not 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 effect while HRESET is asserted. However, the POR sequence is paused if SRESET is asserted during POR.
		Timing	Assertion—May occur at any time, asynchronous to any clock Negation—Must be asserted for at least two *CCB_clk* cycles
READY	O	Ready. Multiplexed with TRIG_OUT. See Chapter 19, "Debug Features and Watchpoint Facility," for more information on TOSR and TRIG_OUT.	
		State Meaning	Asserted—Indicates that the MPC8560 has completed the reset operation and is not in a power-down state (nap, doze or sleep) when TOSR[SEL] equals 0b000. See Section 4.4.2, "Power-On Reset Sequence," for more information.
		Timing	Assertion/Negation—Initial assertion of READY after reset is synchronous with SYSCLK. Subsequent assertion/negation due to power down modes occurs asynchronously.

4.2.2 Clock Signals

Table 4-3 describes the overall clock signals of the MPC8560. Note that some clock signals are specific to blocks within the MPC8560, and although some of their functionality is described in Section 4.4.4, "Clocking," they are defined in detail in their respective chapters.

Note that there is also a CLK_OUT signal in the MPC8560; the signal driven on the CLK_OUT pin is selectable and described in Section 17.4.1.16, "Clock Out Control Register (CLKOCR)."

Table 4-3. 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 for all devices and interfaces that operate synchronously with the core. Multiplied up with a phased-lock loop (PLL) to create the core complex bus (CCB) clock (also called the platform clock) which is used by virtually all of the synchronous system logic, include the L2 cache, the DDR SDRAM and local bus memory controllers, and other internal blocks such as the DMA and interrupt controllers. The CCB clock, in turn, feeds the PLL in the e500 core and the DLLs that create the DDR SDRAM and local bus memory clocks. When the PCI/PCI-X interface is used, SYSCLK also functions as the PCI_CLK signal. Note that this is true whether the MPC8560 is in agent or host mode. The MPC8560 does not provide a separate PCI_CLK output in host mode.	
		Timing	Assertion/Negation—See the *MPC8560 Integrated Processor Hardware Specifications* for specific timing information for this signal.
RTC	I	Real time clock. May be used (optionally) to clock the time base of the e500 core. The maximum input frequency should not exceed 1/8th the core frequency. See Section 4.4.4.4, "Real Time Clock." This signal can also be used (optionally) to clock the global timers in the programmable interrupt controller (PIC).	
		Timing	Assertion/Negation—See the *MPC8560 Integrated Processor Hardware Specifications* for specific timing information for this signal.

4.3 Memory Map/Register Definition

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

4.3.1 Local Configuration Control

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

Table 4-4. Local Configuration Control Register Map

Local Memory Offset (Hex)	Register	Access	Reset	Section/Page
0x0_0000	CCSRBAR—Configuration, control, and status registers base address register	R/W	0x000F_F700	4.3.1.1.2/4-5
0x0_0008	ALTCBAR—Alternate configuration base address register	R/W	0x0000_0000	4.3.1.2.1/4-6
0x0_0010	ALTCAR—Alternate configuration attribute register	R/W	0x0000_0000	4.3.1.2.2/4-6
0x0_0020	BPTR—Boot page translation register	R/W	0x0000_0000	4.3.1.3.1/4-8

4.3.1.1 Accessing Configuration, Control, and Status Registers

The configuration, control, and status registers are memory mapped. The set of configuration, control, and status registers occupies a 1-Mbyte region of memory. Their location is programmable using the CCSR base address register (CCSRBAR). The default base address for the configuration, control, and status registers is 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.

4.3.1.1.1 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 (for example, boot ROM). Follow this load with an **isync**.
– Read the contents of CCSRBAR from its new location, followed by another **isync** instruction.

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

Figure 4-1 shows the fields of CCSRBAR.

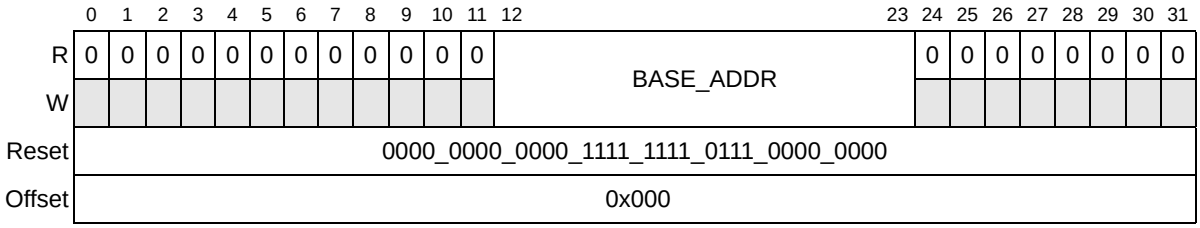


Figure 4-1. Configuration, Control, and Status Register Base Address Register (CCSRBAR)

Table 4-5 defines the bit fields of CCSRBAR.

Table 4-5. CCSRBAR Field Descriptions

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

4.3.1.2 Accessing Alternate Configuration Space

An alternate configuration space can be accessed by configuring the ALTCBAR and ALTCAR registers. These are intended to be used with the boot sequencer to allow the boot sequencer to access an alternate 1-Mbyte region of configuration space. By loading the proper boot sequencer command in the serial ROM the base address in the ALTCBAR can be combined with the 20 bits of address offset supplied from the serial ROM to generate a 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 11.4.5, “Boot Sequencer Mode,” for more information.

NOTE

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

4.3.1.2.1 Alternate Configuration Base Address Register (ALTCBAR)

Figure 4-2 shows the fields of ALTCBAR.

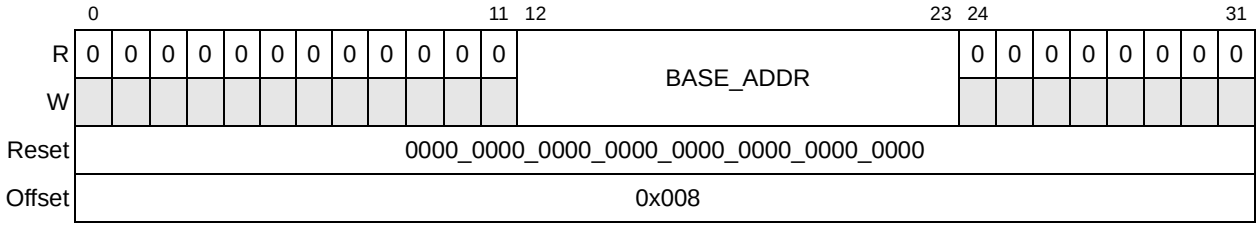


Figure 4-2. Alternate Configuration Base Address Register (ALTCBAR)

Table 4-6 defines the bit fields of ALTCBAR.

Table 4-6. ALTCBAR Field Descriptions

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

4.3.1.2.2 Alternate Configuration Attribute Register (ALTCAR)

Figure 4-3 shows the fields of ALTCAR.

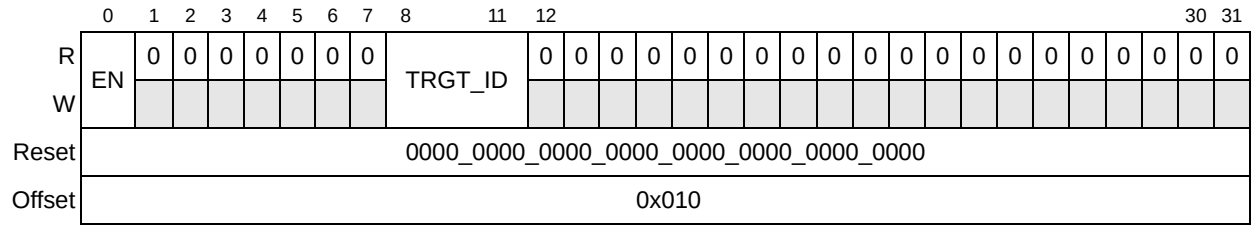


Figure 4-3. Alternate Configuration Attribute Register (ALTCAR)

Table 4-7 defines ALTCAR fields.

Table 4-7. ALTCAR Field Descriptions

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

4.3.1.3 Boot Page Translation

When the e500 core comes out of reset, its MMU has one 4-Kbyte page defined at 0xFFFF_F*nnn*. The first instruction executed by the e500 core is always address 0xFFFF_FFFC, which must be a branch to an address within the 4-Kbyte boot page. 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). If enabled, this translation affects CPU accesses to address range 0xFFFF_F*nnn*.

The boot sequencer can enable boot page translation, or the boot page translation can be enabled by an external host when 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 (8 Mbytes at 0xFF80_0000 to 0xFFFF_FFFF as defined in Section 4.4.3.3, “Boot ROM Location”), 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 BPTR defines only the address translation, not the target interface. See Section 2.1, “Local Memory Map Overview and Example,” and Section 11.4.5, “Boot Sequencer Mode,” for more information.

4.3.1.3.1 Boot Page Translation Register (BPTR)

Figure 4-4 shows the fields of BPTR.

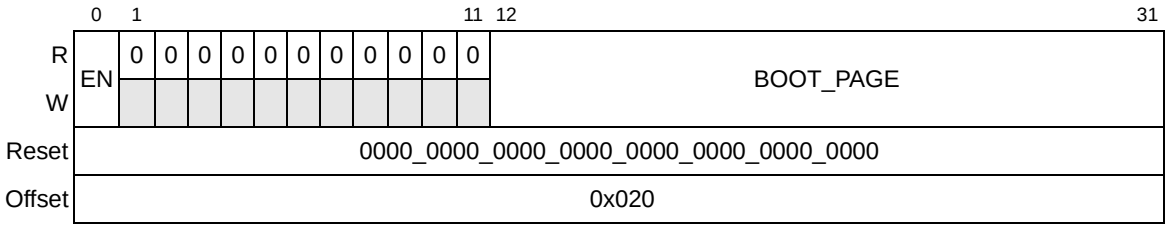


Figure 4-4. Boot Page Translation Register (BPTR)

Table 4-8 describes BPTR field descriptions.

Table 4-8. BPTR Field Descriptions

Bits	Name	Description
0	EN	Boot page translation enable 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_F*nnn* are replaced with this value.

4.3.2 Boot Sequencer

The boot sequencer is a DMA engine that accesses a serial ROM on the I^2C interface and writes data to CCSR memory or the memory space pointed to by the alternate configuration base address register (ALTCBAR). See Section 4.3.1.2, “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 11.4.5, “Boot Sequencer Mode,” in the I^2C chapter.

4.4 Functional Description

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

4.4.1 Reset Operations

The MPC8560 has reset input signals for hard and soft reset operation.

4.4.1.1 Soft Reset

Assertion of the SRESET signal causes the CPM to go through its soft reset sequence. In addition, assertion of 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 *PowerPC™ e500 Core Complex Reference Manual* for more information on the machine check interrupt, and Section 17.4.1.13, "Machine Check Summary Register (MCPSUMR)," for more information on the setting of the soft reset flag. Note that if SRESET is asserted before the e500 core is configured to handle a machine check interrupt, a core checkstop condition occurs, which causes CKSTP_OUT to assert.

4.4.1.2 Hard Reset

The MPC8560 can be completely reset by the assertion of the HRESET input. The assertion of this signal by external logic is the equivalent of a POR and causes the sequence of events described in Section 4.4.2, "Power-On Reset Sequence."

Refer to the *MPC8560 Integrated Processor Hardware Specifications* for the timing requirements for HRESET assertion and negation.

The MPC8560 hard reset request output signal (HRESET_REQ) indicates to external logic that a hard reset is being requested by hardware or a RapidIO device. Hardware causes this signal to assert for a boot sequencer failure (see Section 11.4.5, "Boot Sequencer Mode," and Section 11.4.5.2, "EEPROM Data Format,") or when the e500 watchdog timer is configured to cause a reset request when it expires. A RapidIO device causes this signal to assert if it sends four consecutive RapidIO link maintenance reset commands without any other intervening packets or control symbols, except idle control symbols (see Section 16.4.1, "RapidIO Transaction, Packet, and Control Symbol Summary," for details of RapidIO transaction types).

4.4.2 Power-On Reset Sequence

The POR sequence for the MPC8560 is as follows:

1. Power is applied to meet the specifications in the *MPC8560 Integrated Processor Hardware Specifications*.
2. System asserts HRESET and TRST causing all registers to be initialized to their default states and most I/O drivers to be three-stated (some clock, clock enabled, and system control signals are active).
3. System applies a stable SYSCLK signal and stable PLL configuration inputs, and the device PLL begins locking to SYSCLK.
4. System negates HRESET after its required hold time and after POR configuration inputs have been valid for at least 4 SYSCLK cycles.

NOTE

If the JTAG signals are not used, $\overline{\text{TRST}}$ 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. The MPC8560 enables I/O drivers.
6. The MPC8560 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 e500and CPM PLL configuration 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 50 µs 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 DLLs and other remaining I/O blocks. The DLLs 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. The ASLEEP signal negates synchronized to a rising edge of SYSCLK, indicating the ready state. The ready state is also indicated by the assertion of READY/TRIG_OUT if TOSR[SEL] = 000. In this case, READY is asserted with the same rising edge of SYSCLK, to indicate that the device has reached its ready state. See Section 19.3.4.1, “Trigger Out Source Register (TOSR),” for more information on this register.

    Asserting READY allows external system monitors to know basic device status. For example, exactly when it emerges from reset, or if the device is in a low-power mode. For more information on the debug functions of TRIG_OUT, see Section 19.3.4, “Trigger Out Function.” For more information about power management states, see Section 17.4.1, “Register Descriptions.”

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

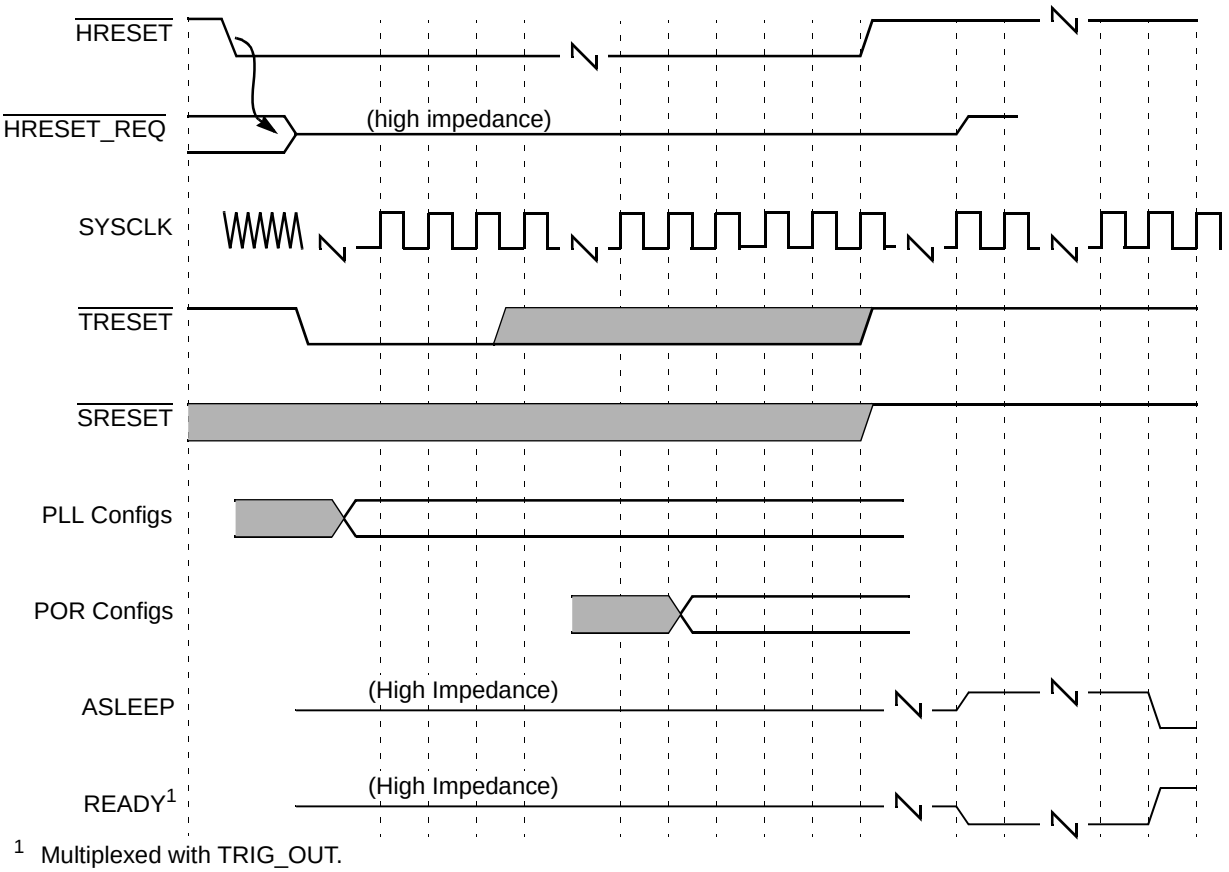


Figure 4-5. Power-On Reset Sequence

4.4.3 Power-On Reset Configuration

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

Most POR configuration signals have internal pull-up resistors so that if the desired setting is high, there is no need for a pull-up resistor on the board. Other POR configuration signals do not use pull-ups and, therefore, must be pulled high or low. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper resistor values to be used for pulling POR configuration signals high or low.

This section describes the functions and modes configured by POR configuration signals. Note that many reset configuration settings are accessible to software through the following read-only memory-mapped registers described in Chapter 17, “Global Utilities.”

- POR PLL status register (PORPLLSR)
- POR boot mode status register (PORBMSR)
- POR I/O impedance status and control register (PORIMPSCR)
- POR device status register (PORDEVSR)
- POR debug mode status register (PORDBGMSR)
- General-purpose POR configuration register (GPPORCR)—reports the value on LAD[0:31] during POR (can be used to external system configuration)

NOTE

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

4.4.3.1 System PLL Ratio

The system PLL inputs, shown in Table 4-9, establish the clock ratio between the PCI_CLK/SYSCLK input and the platform clock used by the MPC8560. The platform clock, also called the CCB clock, drives the L2 cache, the DDR SDRAM data rate, and the e500 core complex bus (CCB). There is no default value for this PLL ratio; these signals must be pulled to the desired values. Note that the values latched on these signals during POR are accessible in the PORPLLSR (POR PLL status register), as described in Section 17.4.1.1, “POR PLL Status Register (PORPLLSR).”

Table 4-9. CCB Clock PLL Ratio

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

4.4.3.2 e500 Core PLL Ratio

Table 4-10 describes the e500 core clock PLL inputs that program the core PLL and establish the ratio between the e500 core clock and the e500 core complex bus (CCB) clock. There is no default value for this PLL ratio; these signals must be pulled to the desired values. Note that the values latched on these signals during POR are accessible through the memory-mapped PORPLLSR, as described in Section 17.4.1.1, “POR PLL Status Register (PORPLLSR),” and also in the e500 core HID1 register, as described in Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

Table 4-10. e500 Core Clock PLL Ratios

Functional Signals	Reset Configuration Name	Value (Binary)	e500 Core: CCB ClockRatio
LALE, LGPL2 No default	cfg_core_pll[0:1]	00	2 : 1
		01	5 : 2 (2.5:1)
		10	3 : 1
		11	7 : 2 (3.5:1)

4.4.3.3 Boot ROM Location

The first instruction executed by the e500 core is always address 0xFFFF_FFFC, which must be a branch to an address within the 4-Kbyte boot page. The MPC8560 defines the default boot ROM address range to be 8 Mbytes at address 0xFF80_0000 to 0xFFFF_FFFF. However, which on-chip peripheral handles these boot ROM accesses can be selected at power up.

The boot ROM location inputs, shown in Table 4-11, 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.

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 17.4.1.2, "POR Boot Mode Status Register (PORBMSR)."

Table 4-11. Boot ROM Location

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
TSEC1_TXD[6:4] 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)

See Section 2.1, "Local Memory Map Overview and Example," for an example memory map that relies on the default boot ROM values. Also, see Section 4.3.1.3.1, "Boot Page Translation Register (BPTR)," for information on translation of the boot page. If enabled, this translation only affects CPU accesses to 0xFFF_F*nnn*.

4.4.3.4 Host/Agent Configuration

The host/agent reset configuration inputs, shown in Table 4-12, 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/PCI-X Bus Interface," and Chapter 16, "RapidIO Interface," respectively.

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 17.4.1.2, “POR Boot Mode Status Register (PORBMSR).”

Table 4-12. 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

The CPU boot configuration input, shown in Table 4-13, 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 8.2.1.2, “ECM CCB Port Configuration Register (EEBPCR),” for more information.

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 17.4.1.2, “POR Boot Mode Status Register (PORBMSR).”

Table 4-13. 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 (default).

4.4.3.6 Boot Sequencer Configuration

The boot sequencer configuration options, shown in Table 4-14, 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 11.4.5, "Boot Sequencer Mode," for more information on the boot sequencer.

Note that the values latched on these signals during POR are accessible through the memory-mapped PORBMSR (POR boot mode status register) described in Section 17.4.1.2, "POR Boot Mode Status Register (PORBMSR)."

Table 4-14. 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 (default).

NOTE

When the boot sequencer is enabled, 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

The TSEC width input, shown in Table 4-15, selects standard versus reduced width for both of the three-speed Ethernet controller interfaces. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 17.4.1.4, "POR Device Status Register (PORDEVSR)."

Table 4-15. 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 four transmit data signals and four receive data signals.
		1	Ethernet interfaces operate in their standard TBI or GMII modes using eight transmit data signals and eight receive data signals (default).

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

The TSEC1 protocol input, shown in Table 4-16, selects the protocol (GMII or TBI) used by the TSEC1 controller. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 17.4.1.4, “POR Device Status Register (PORDEVSR).”

Table 4-16. TSEC1 Protocol Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC1_TXD7 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”) (default).

4.4.3.9 TSEC2 Protocol

The TSEC2 protocol input, shown in Table 4-17, selects the protocol (GMII or TBI) used by the TSEC2 controller. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR (POR device status register) described in Section 17.4.1.4, “POR Device Status Register (PORDEVSR).”

Table 4-17. TSEC2 Protocol Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
TSEC2_TXD7 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”) (default).

4.4.3.10 RapidIO Transmit Clock Source

The RapidIO transmit clock source inputs, shown in Table 4-18, specify the source for the RapidIO transmit clock. See Section 4.4.4.2, “RapidIO Clocks,” for more information. Note that the value latched on this signal during POR is accessible through the memory-mapped

PORDEVSR (POR device status register) described in Section 17.4.1.4, “POR Device Status Register (PORDEVSR).”

Table 4-18. 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 inputs (RIO_TX_CLK_IN and RIO_TX_CLK) are 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

The RapidIO device ID inputs, shown in Table 4-19, specify the 3 lower-order bits of the device ID of the MPC8560 as used by RapidIO hosts. Note that the 5 high-order RapidIO device ID bits cannot be set via POR configuration inputs. They may be initialized via the boot sequencer or by the processor from boot ROM or by the RapidIO discovery process.

If configured as a RapidIO host, the upper order device ID bits default to zeros. If configured as a RapidIO agent, the upper order device ID bits default to ones. Unconnected cfg_dev_ID*n* inputs default to 1s regardless of the host/agent mode configuration.

Note that the value latched on this signal at POR is accessible through the memory-mapped PORDEVSR described in Section 17.4.1.4, “POR Device Status Register (PORDEVSR).”

Table 4-19. RapidIO Device ID

Functional Signals	Reset Configuration Name	Meaning
TSEC2_TXD2	cfg_dev_ID5	Device ID used for RapidIO hosts
TSEC2_TXD3	cfg_dev_ID6	Device ID used for RapidIO hosts
TSEC2_TXD4	cfg_dev_ID7	Device ID used for RapidIO hosts

4.4.3.12 PCI Width Configuration

The PCI width configuration input, shown in Table 4-20, configures the PCI/PCI-X interface to 32- or 64-bit extended mode of operation. Note that the value latched on this signal during POR is accessible through the PORDEVSR described in Section 17.4.1.4, “POR Device Status Register (PORDEVSR).”

Table 4-20. 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 (default).

4.4.3.13 PCI I/O Impedance

The PCI I/O impedance input, shown in Table 4-21, selects the impedance of the PCI I/O drivers. Note that the values latched on these signals during POR are accessible through PORIMPSCR, described in Section 17.4.1.3, "POR I/O Impedance Status and Control Register (PORIMPSCR)."

Table 4-21. PCI I/O Impedance

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{PCI\_GNT1}}$	cfg_pci_impd	0	25-Ω I/O drivers are used on the PCI interface.
Default (1)		1	42-Ω I/O drivers are used on the PCI interface (default).

4.4.3.14 PCI Arbiter Configuration

The PCI arbiter configuration input, shown in Table 4-22, enables the on-chip PCI/PCI-X arbiter. Note that the value latched on this signal during POR is accessible through the PORDEVSR described in Section 17.4.1.4, "POR Device Status Register (PORDEVSR)."

Table 4-22. 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 (default).

4.4.3.15 PCI Debug Configuration

The PCI debug configuration input, shown in Table 4-23, enables PCI/PCI-X debug mode. In this mode, source ID information is driven onto the highest order address bits PCI_AD[62:58] during the bus command phase (PCI) or attribute phase (PCI-X). Note that the value latched on this signal during POR is accessible through the PORDBGMSR described in Section 17.4.1.5, "POR Debug Mode Status Register (PORDBGMSR)."

Table 4-23. 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[62:58], during the bus command phase (PCI) or attribute phase (PCI-X).
		1	PCI operates in normal mode (default)

4.4.3.16 PCI-X Configuration

The PCI-X configuration input, shown in Table 4-24, 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. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDEVSR, described in Section 17.4.1.4, "POR Device Status Register (PORDEVSR)."

Table 4-24. 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 (default)

4.4.3.17 Memory Debug Configuration

The memory debug configuration input, shown in Table 4-25, selects which debug outputs (DDR or LBC memory controller) are driven onto the MSRCID and MDVAL debug signals. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDBGMSR (POR debug mode register) described in Section 17.4.1.5, "POR Debug Mode Status Register (PORDBGMSR)."

Table 4-25. 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 SDRAM controller is driven on the MSRCID and MDVAL signals (default).

4.4.3.18 DDR Debug Configuration

The DDR debug configuration input, shown in Table 4-26, enables a DDR memory controller debug mode in which the DDR SDRAM source ID field and data valid strobe are driven onto the

ECC pins. ECC checking and generation are disabled in this case. ECC signals driven from the SDRAMs must be electrically disconnected from the ECC I/O pins of the MPC8560 in this mode. Note that the value latched on this signal during POR is accessible through the memory-mapped PORDBGMSR (POR debug mode register) described in Section 17.4.1.5, "POR Debug Mode Status Register (PORDBGMSR)."

Table 4-26. 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. ECC signals from memory devices must be disconnected.
		1	Debug information is not driven on ECC pins. ECC pins function in their normal mode (default).

4.4.3.19 PCI/PCI-X Output Hold Configuration

The PCI output hold configuration inputs configure the output hold times for the PCI or PCI-X output drivers. The meanings are different for PCI and PCI-X because the required hold times are different in the two specifications. In either mode, the default value will meet the hold times required by the respective specification. Hold times are adjusted by adding or subtracting buffer delays to the intrinsic delay of the output driver. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for specific timing information. Table 4-27 shows the hold time configurations for PCI mode, and Table 4-28 shows the hold time configurations for PCI-X mode.

Table 4-27. 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	Two added buffer delays—required to meet 2-ns hold time requirement
		10	Three added buffer delays (default + 1)
		01	Zero added buffer delays (default + 2 mod 4)
		00	One added buffer delay (default + 3 mod 4)

Table 4-28. 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	Zero added buffer delays—meets 0.7 ns hold time requirement
		10	One added buffer delay (default + 1)
		01	Two added buffer delays (default + 2)
		00	Three added buffer delays (default + 3)

4.4.3.20 Local Bus Output Hold Configuration

The LBC output hold configuration inputs, shown in Table 4-29, configure the output hold times for the local bus interface output drivers. The default values are designed to meet the specified AC timing requirements for the local bus. Hold times are adjusted by adding buffer delays to the intrinsic delay of the output driver. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for specific timing information. Note that for three of the POR configuration input settings, there is a minimum of one buffer of output hold delay between the LALE and LAD[0:31] signals.

Table 4-29. Local Bus Output Hold Configuration

Functional Signals	Reset Configuration Name	Value (Binary)	Meaning
TSEC2_TXD[6:5] Default (11)	cfg_lb_hold[0:1]	11	One added buffer delay (default) (zero added buffer delays for LALE)
		10	Two added buffer delays (default + 1) (one added buffer delay for LALE)
		01	Three added buffer delays (default + 2) (one added buffer delay for LALE)
		00	Zero added buffer delays (zero added buffer delays for LALE)

4.4.3.21 General-Purpose POR Configuration

The LBC address/data bus inputs, shown in Table 4-30, configure the value of the general-purpose POR configuration register defined in Section 17.4.1.6, “General-Purpose POR Configuration Register (GPPORCR).” This register is intended to facilitate POR configuration of user systems. A value placed on LAD[0:31] during POR is captured and stored (read only) in the GPPORCR. Software can then use this value to inform the operating system about initial system configuration. Typical interpretations include circuit board type, board ID number, or a list of available peripherals.

Table 4-30. General-Purpose POR Configuration

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

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. As shown in Figure 4-6, the SYSCLK input (frequency) is multiplied up using a phase lock loop (PLL) to create the core complex bus (CCB) clock (also called the platform clock). The CCB clock is used by virtually all of the synchronous system logic, including the L2 cache, and other internal blocks such as the DMA and interrupt controller. The CCB clock also feeds the PLL in the e500 core and the DLLs that create clocks for the DDR SDRAM and local bus memory controllers. Note that the divide-by-two CCB clock divider and the divide-by-*n* CCB clock divider, shown in Figure 4-6, are located in the DDR and local bus blocks, respectively.

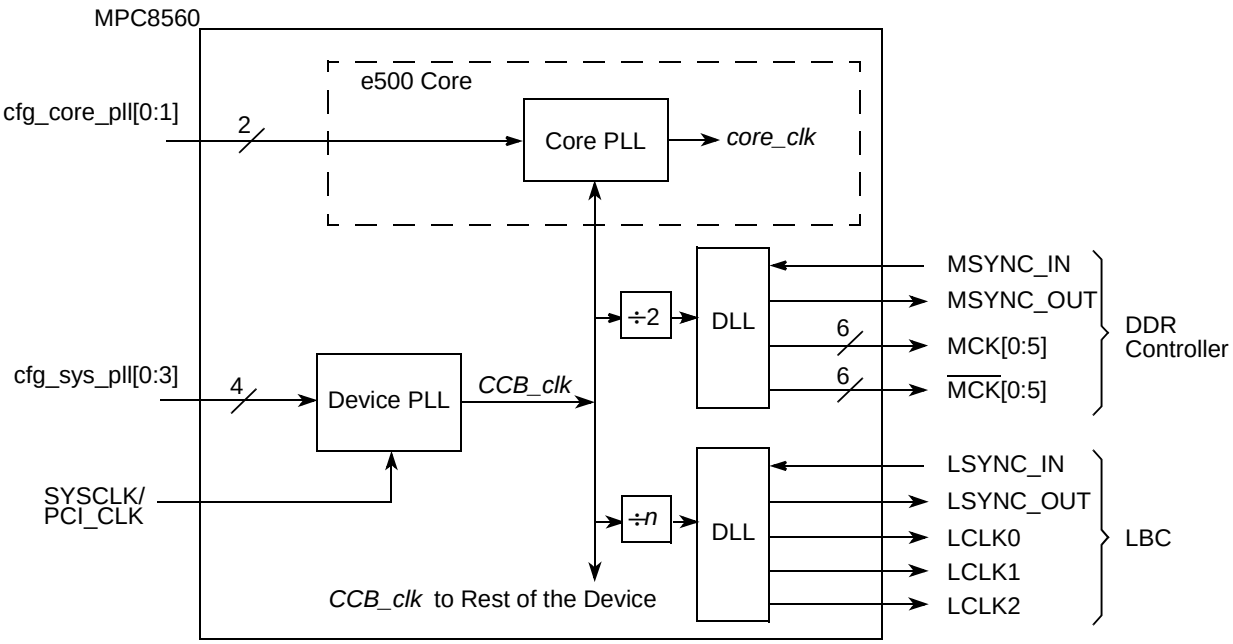


Figure 4-6. Clock Subsystem Block Diagram

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 host mode. The MPC8560 does not provide a separate PCI_CLK output in host mode.

4.4.4.2 RapidIO Clocks

As shown in Figure 4-7, 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 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 $\overline{\text{RIO\_RCLK}}$).

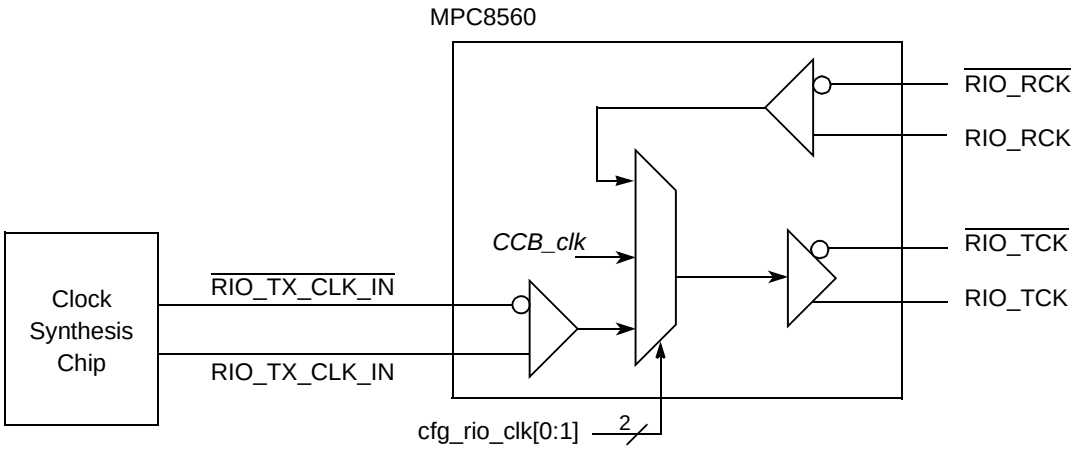


Figure 4-7. RapidIO Transmit Clock Options

If the MPC8560 is 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 dedicated LVDS clock input, RIO_TX_CLK_IN / $\overline{\text{RIO\_TX\_CLK\_IN}}$.

4.4.4.3 Ethernet Clocks

The Ethernet blocks operate asynchronously with respect to the rest of the device. These blocks use receive and transmit clocks supplied by their respective PHY chips, 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

As shown in Figure 4-8, the real time clock (RTC) input can optionally be used to clock the e500 core timer facilities. RTC can also be used (optionally) by the MPC8560 programmable interrupt controller (PIC) global timer facilities. The RTC is separate from the e500 core clock and is intended to support relatively low frequency timing applications. The RTC frequency range is specified in the *MPC8560 Integrated Processor Hardware Specifications*, but the maximum value should not exceed one-eighth of the core frequency.

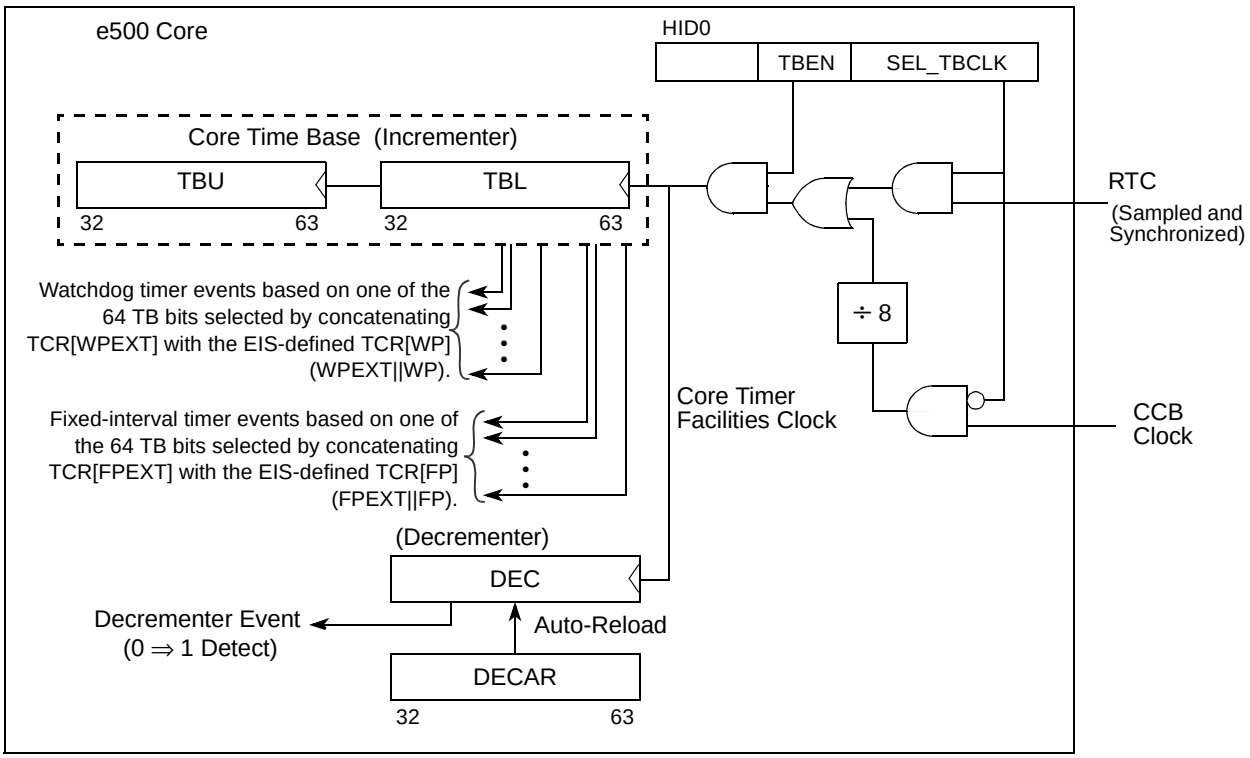
Before being distributed to the core time base, RTC is sampled and synchronized with the CCB clock.

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

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

The default source of the time base is the CCB clock divided by eight. For more details, see the section “Hardware Implementation-Dependent Register (HID0),” in the “*PowerPC™ e500 Core Complex Reference Manual.*”

Section 10.3.2.6, “Timer Control Register (TCR),” provides additional information on the use of the RTC signal to clock the global timers in the PIC unit.



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

Figure 4-8. RTC and Core Timer Facilities Clocking Options

Part II
e500 Core Complex and L2 Cache

This part describes the many features of the MPC8560 core processor at an overview level and the interaction between the core complex and the L2 cache. The following chapters are included:

- Chapter 5, “Core Complex Overview,” provides an overview of the e500 core processor and the L1 caches and MMU that, together with the core, comprise the core complex.
- Chapter 6, “Core 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 through the core complex bus (CCB).

Chapter 5
Core Complex Overview

This chapter provides an overview of the PowerPC e500 microprocessor core as it is implemented on the MPC8560.

This chapter 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)
- A summary of the Book E architecture compatibility and migration from the original version of the PowerPC architecture as it is defined by Apple, IBM, and Motorola (referred to as the AIM version of the PowerPC architecture)

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

5.1 Overview

The e500 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 is a 32-bit implementation of the Book E architecture using the lower words in the 64-bit general-purpose registers (GPRs).

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

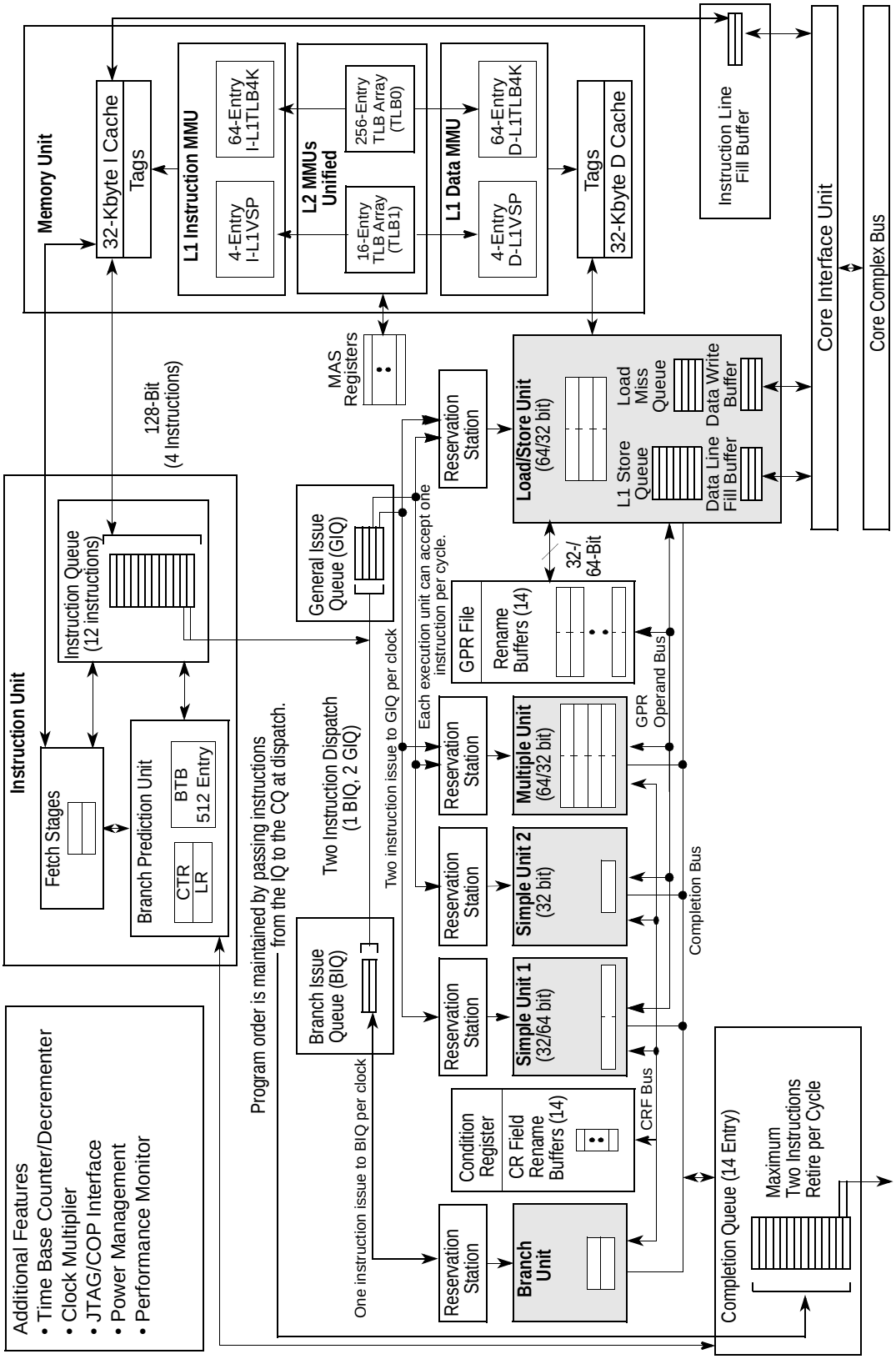


Figure 5-1. e500 Core Complex Block Diagram

Book E allows processors to provide auxiliary processing units (APUs), which are extensions to the architecture that can perform computational or system management functions. One 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 Freescale Semiconductor Book E implementation standards (EIS).

5.1.1 Upward Compatibility

The e500 provides 32-bit effective addresses and integer data types of 8, 16, and 32 bits, and two-element 64-bit data types for the embedded vector floating-point APU which provides scalar and vector single-precision instructions that operate on operands comprised of two 32-bit elements and the SPE APU, which provides an extensive instruction set for 64-bit vector integer and fractional operations.

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

NOTE

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

Freescale Semiconductor offers a libmoto_e500 library that uses SPE and embedded floating-point APU instructions. Freescale will also provide libraries to support next-generation PowerQUICC devices.

5.1.2 Core Complex Summary

The core complex is a superscalar processor that can issue two instructions and complete two instructions per clock cycle. Instructions complete in order, but can execute out of order. Execution results are available to subsequent instructions through the rename buffers, but those results are recorded into architected registers in program order, maintaining a precise exception model. All arithmetic instructions that execute in the core operate on data in the GPRs. Although the GPRs are 64 bits wide, only SPE APU and embedded vector floating-point instructions operate on the upper word of the GPRs; the upper 32 bits 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 LSU and SU2 support 64- and 32-bit instructions.

The ability to execute five instructions in parallel and the use of simple instructions with short execution times yield high efficiency and throughput. Most integer instructions execute in 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. These arrays are maintained completely by the hardware with a true least-recently-used (LRU) algorithm.
- The second-level MMU contains a 16-entry, fully-associative unified (instruction and data) TLB array that provides support for variable-sized pages 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 e500 Processor and System Version Numbers

Table 5-1 matches the revision code in the processor version register (PVR) and the system version register (SVR) to the revision level marked on the device. These registers can be accessed as SPRs through the e500 core (see Chapter 6, “Core Register Summary”) or as memory-mapped registers defined by the integrated device (see Section 18.4.1, “Register Descriptions”).

Table 5-1. Revision Level-to-Device Marking Cross-Reference

MPC8560 Revision	e500 Core Revision	Processor Version Register (PVR)	System Version Register (SVR)
1.0	1.0	0x8020_0010	0x8070_0010
2.0	2.0	0x8021_0010	0x8070_0020

5.3 Features

Key features of the e500 are summarized as follows:

- Implements Book E 32-bit architecture
- Auxiliary processing units

  The branch target buffer (BTB) locking APU is specific to the e500. The BTB locking APU gives the user the ability to lock, unlock, and invalidate BTB entries. The EIS defines the following APUs:

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

  — Cache line 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 5.8, “Interrupts and Exception Handling.”

  — Single-precision embedded scalar and vector floating-point APUs. These instructions are listed in the *EREF: A Reference for Freescale Semiconductor Book E and the e500 Core*.

  — 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 64-bit loads and stores are executed in the common LSU. Figure 5-1 shows how execution logic for SU1, the MU, and the LSU is replicated to support operations on the upper halves of the GPRs.

The e500 register set is modified as follows:

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

These registers are shown in Figure 5-6. 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
– 64-bit loads and stores
– 64-bit merge instructions

- Cache structure—Separate 32-Kbyte, 32-byte line, 8-way set-associative level 1 instruction and data caches
  — 1.5-cycle cache array access, 3-cycle load-to-use latency
  — Pseudo-LRU (PLRU) replacement algorithm
  — Copy-back data cache that can function as a write-through cache on a page-by-page basis
  — Supports all Book E memory coherency modes
  — Supports EIS-defined cache-locking instructions, as listed in Table 5-3
- Dual-issue superscalar control
  — Two-instructions-per-clock peak issue rate
  — Precise exception handling

- Decode unit
  - 12-entry instruction queue (IQ)
  - Full hardware detection of interlocks
  - Decodes as many as two instructions per cycle
  - Decode serialization control
  - Register dependency resolution and renaming
- Branch prediction unit (BPU)
  - Dynamic branch prediction using a 512-entry, four-way set-associative branch target buffer (BTB) supported by the e500 BTB instructions listed in Table 5-5.
  - Branch prediction is handled in the fetch stages.
- Completion unit
  - As many as 14 instructions allowed in 14-entry completion queue (CQ)
  - In-order retirement of as many as two instructions per cycle
  - Completion and refetch serialization control
  - Synchronization for all instruction flow changes—interrupts, mispredicted branches, context-synchronizing instructions
- Issue queues
  - Two-entry branch instruction issue queue (BIQ)
  - Four-entry general instruction issue queue (GIQ)
- Branch unit—The branch unit (BU) is an execution unit and is distinct from the BPU. It executes (resolves) all branch and CR logical instructions.
- Two simple units (SU1 and SU2)
  - Add and subtract
  - Shift and rotate
  - Logical operations
  - Support for 64-bit signal processing engine APU instructions in SU1
- Multiple-cycle unit (MU)—The MU is shown in Figure 5-2.

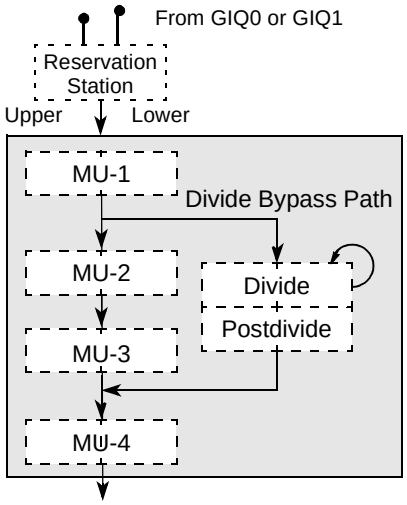


Figure 5-2. Four-Stage MU Pipeline, Showing Divide Bypass

The MU has the following features:

— Four-cycle latency for all multiplication, including SPE integer and fractional multiply instructions and embedded scalar and vector floating-point multiply instructions

— Variable-latency divide: 4, 11, 19, and 35 cycles for all integer divide instructions. If **r**A or **r**B is zero, floating-point divide instructions take 4 cycles. All others take 29. Note that although most divide instructions take more than 4 cycles to execute, the MU allows subsequent multiply instructions to execute through all four MU stages in parallel with the divide.

— Four-cycle floating-point add and subtract

- Load/store unit (LSU) is shown in Figure 5-3.

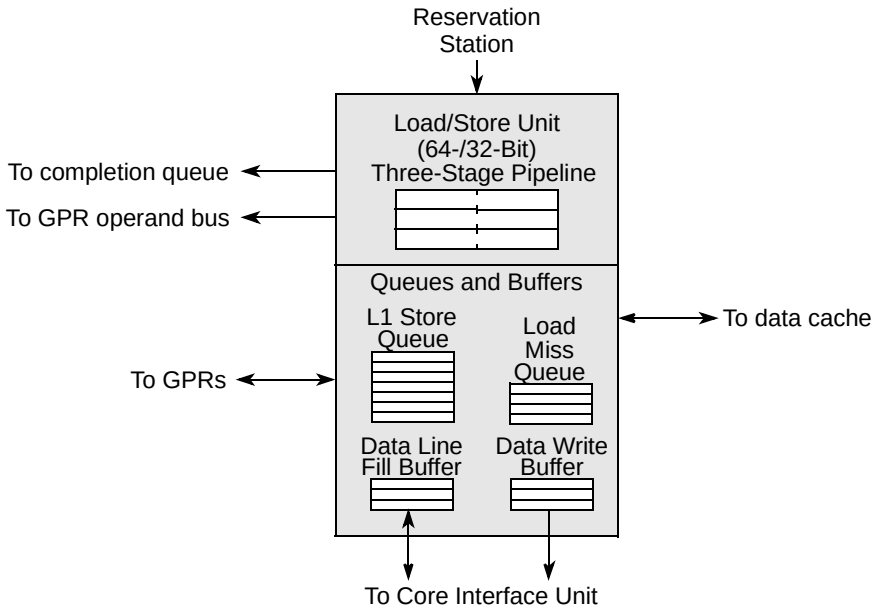


Figure 5-3. Three-Stage Load/Store Unit

The LSU has the following features:

— 3-cycle load latency

— Fully pipelined

— The 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.

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

- Cache coherency
  — Supports four-state cache coherency: modified-exclusive, exclusive, shared, and invalid (MESI)
  — Bus support for hardware-enforced coherency (bus snooping)
- Core complex bus (CCB)—internal bus
  — High-speed, on-chip local bus with data tagging
  — 32-bit address bus
  — Address protocol with address pipelining and retry/copyback derived from bus used by previous generations of PowerPC processors (referred to as the 60x bus)
  — Two general-purpose read data buses and one write data bus

- 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
    - Machine-check APU
    - SPE APU unavailable exception
    - Floating-point data exception
    - Floating-point round exception
    - Performance monitor
- Memory management unit (MMU)
  - 32-bit effective address translated to 32-bit real address (using a 41-bit interim virtual address)
  - TLB entries for variable (4-Kbyte–256-Mbyte) and fixed-size (4-Kbyte) pages
  - Data L1 MMU
    - 4-entry, fully associative TLB array for variable-sized pages
    - 64-entry, 4-way set-associative TLB for 4-Kbyte pages
  - Instruction L1 MMU
    - 4-entry, fully associative TLB array for variable-sized pages
    - 64-entry, 4-way set-associative TLB for 4-Kbyte pages
  - Unified L2 MMU
    - 16-entry, fully associative TLB array for variable-sized pages
    - A 256-entry, 2-way set-associative unified (for instruction and data accesses) L2 TLB array (TLB0) supports only 4-Kbyte pages
  - Software reload for TLBs
  - Virtual memory support for as much as 4 Gbytes (2^{32}) of effective address space
  - Real memory support for as much as 4 Gbytes (2^{32}) of physical memory
  - Support for big-endian and true little-endian memory on a per-page basis

- Power management
  - Low-power, 1.5-V design
  - Internal clock multipliers ranging from 1 to 8 times the bus clock, including integer and half-mode multipliers. The MPC8560 supports multipliers of 2, 2.5, 3, and 3.5.
  - Power-saving modes: core-halted and core-stopped
  - Dynamic power management of execution units, caches, and MMUs
  - NAP, DOZE, and SLEEP bits in HID0 can be used to assert *nap*, *doze*, and *sleep* output signals to initiate power-saving modes at the integrated device level.
- Testability
  - LSSD scan design
  - JTAG interface
  - ESP support
  - ABIST for arrays
  - LBIST
- Reliability and serviceability
  - Parity checking on caches
  - Parity checking on e500 local bus

5.4 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 Book E floating-point, load string, or store string instructions.
- The e500 supports the following implementation-specific instructions:
  - Integer select APU. This APU consists of the Integer Select instruction (**isel**), which functions as an if-then-else statement that selects between two source registers by comparison to a CR bit. This instruction eliminates conditional branches, decreases band latency, and reduces the code footprint.
  - Performance monitor APU. Table 5-2 lists performance monitor APU instructions.

Table 5-2. 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 line lock and unlock APU. The cache block lock and unlock APU consists of the instructions described in Table 5-3.

Table 5-3. 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

— Machine check APU. This APU defines the Return from Machine Check Interrupt instruction (**rfmci**).

— 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). Some scalar instructions are defined for DSP that produce a 64-bit scalar result.

— The embedded floating-point APUs provide single-precision scalar and vector floating-point instructions. Scalar floating-point instructions use only the lower 32 bits of the GPRs for single-precision floating-point calculations. Table 5-4 lists embedded floating-point instructions.

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

Instruction	Mnemonic		Syntax
	Scalar	Vector	
Convert Floating-Point from Signed Fraction	**efscfsf**	**evfscfsf**	**r**D,**r**B
Convert Floating-Point from Signed Integer	**efscfsi**	**evfscfsi**	**r**D,**r**B
Convert Floating-Point from Unsigned Fraction	**efscfuf**	**evfscfuf**	**r**D,**r**B
Convert Floating-Point from Unsigned Integer	**efscfui**	**evfscfui**	**r**D,**r**B
Convert Floating-Point to Signed Fraction	**efsctsf**	**evfsctsf**	**r**D,**r**B
Convert Floating-Point to Signed Integer	**efsctsi**	**evfsctsi**	**r**D,**r**B
Convert Floating-Point to Signed Integer with Round toward Zero	**efsctsiz**	**evfsctsiz**	**r**D,**r**B
Convert Floating-Point to Unsigned Fraction	**efsctuf**	**evfsctuf**	**r**D,**r**B
Convert Floating-Point to Unsigned Integer	**efsctui**	**evfsctui**	**r**D,**r**B
Convert Floating-Point to Unsigned Integer with Round toward Zero	**efsctuiz**	**evfsctuiz**	**r**D,**r**B
Floating-Point Absolute Value	**efsabs**	**evfsabs**	**r**D,**r**A
Floating-Point Add	**efsadd**	**evfsadd**	**r**D,**r**A,**r**B

Table 5-4. Scalar and Vector Embedded Floating-Point APU Instructions (continued)

Instruction	Mnemonic		Syntax
	Scalar	Vector	
Floating-Point Compare Equal	**efscmpeq**	**evfscmpeq**	**cr**D,**r**A,**r**B
Floating-Point Compare Greater Than	**efscmpgt**	**evfscmpgt**	**cr**D,**r**A,**r**B
Floating-Point Compare Less Than	**efscmplt**	**evfscmplt**	**cr**D,**r**A,**r**B
Floating-Point Divide	**efsdiv**	**evfsdiv**	**r**D,**r**A,**r**B
Floating-Point Multiply	**efsmul**	**evfsmul**	**r**D,**r**A,**r**B
Floating-Point Negate	**efsneg**	**evfsneg**	**r**D,**r**A
Floating-Point Negative Absolute Value	**efsnabs**	**evfsnabs**	**r**D,**r**A
Floating-Point Subtract	**efssub**	**evfssub**	**r**D,**r**A,**r**B
Floating-Point Test Equal	**efststeq**	**evfststeq**	**cr**D,**r**A,**r**B
Floating-Point Test Greater Than	**efststgt**	**evfststgt**	**cr**D,**r**A,**r**B
Floating-Point Test Less Than	**efststlt**	**evfststlt**	**cr**D,**r**A,**r**B

— 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, organized as 128 rows with four-way set associativity, that holds the address and target instruction of the 512 most-recently taken branches. Table 5-5 lists BTB instructions.

Table 5-5. BTB Locking APU Instructions

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

5.5 Instruction Flow

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

A superscalar processor is one that issues multiple independent instructions into separate execution units, allowing parallel execution. The e500 core has five execution units, one each for branch (BU), load/store (LSU), and multiple-cycle operations (MU), and two for simple arithmetic

(SU1 and SU2). The MU and SU1 arithmetic execution units also execute 64-bit SPE vector instructions, using both the lower and upper halves of the 64-bit GPRs.

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

5.5.1 Initial Instruction Fetch

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

5.5.2 Branch Detection and Prediction

To improve branch performance, the e500 provides an implementation-specific dynamic branch prediction using the BTB to resolve branch instructions and improve the accuracy of branch predictions. Each of the 512 entries in the four-way set associative address cache of branch target addresses includes a 2-bit saturating branch history counter, whose value is incremented or decremented depending on whether the branch was taken. These bits can take four values indicating strongly taken, weakly taken, weakly not taken, and strongly not taken.

The BTB is used not only to predict branches, but to detect branches during the fetch stage, offering an efficient way to access instruction streams for branches predicted as taken.

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

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

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

5.5.3 e500 Execution Pipeline

The seven stages of the e500 execution pipeline—fetch1, fetch2/predecode, decode/dispatch, issue, execute, complete, and write back—are highlighted in grey in Figure 5-4.

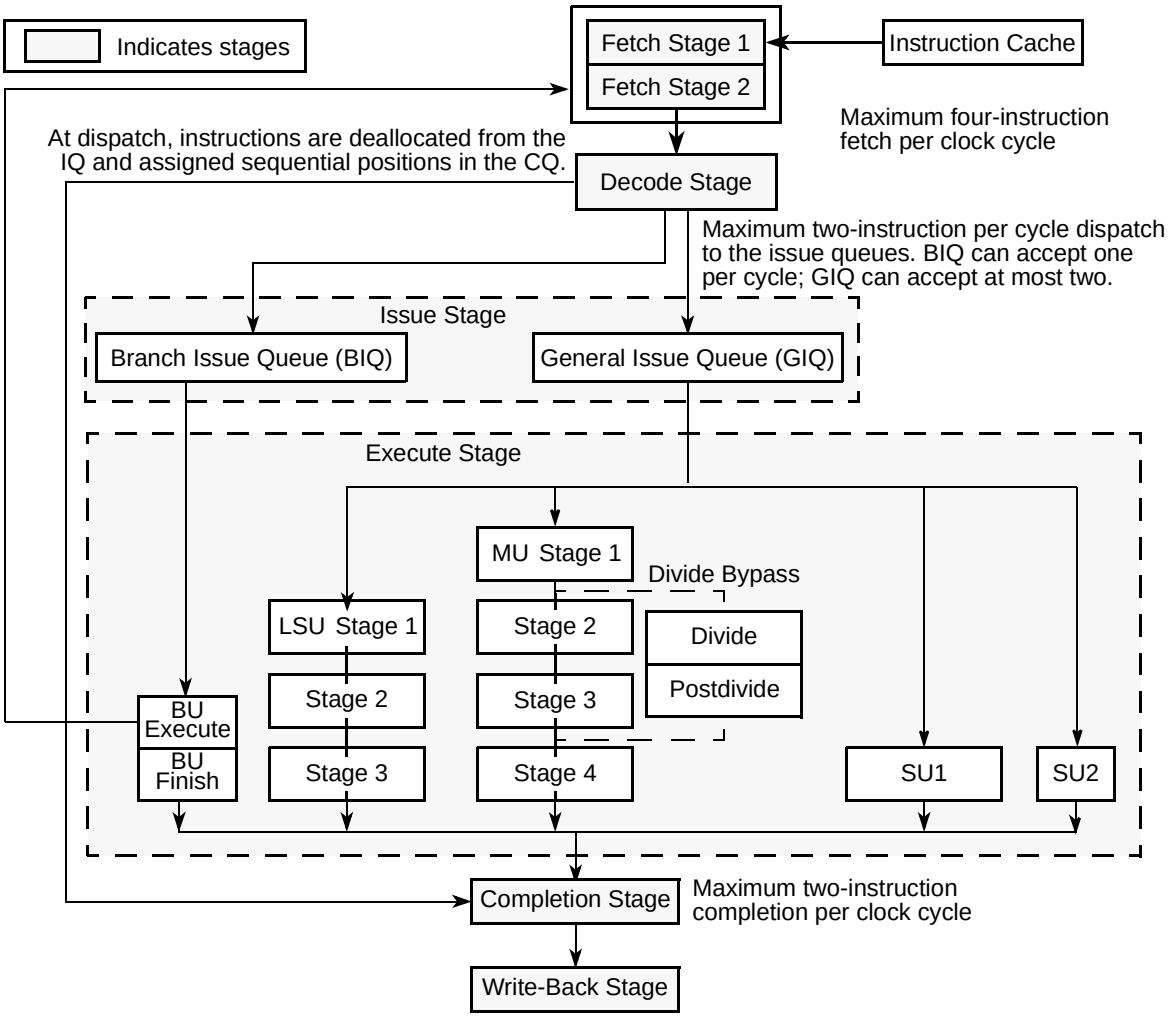


Figure 5-4. Instruction Pipeline Flow

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

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

  The general behavior of the two issue queues is described as follows:
  - The GIQ accepts as many as two instructions from the dispatch unit per cycle. SU1, SU2, MU, and all LSU instructions (including 64-bit loads and stores) are dispatched to the GIQ, shown in Figure 5-5.

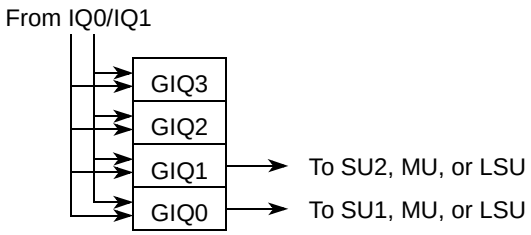


Figure 5-5. GPR Issue Queue (GIQ)

    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.

Branches resolve in execute stage. If a branch is mispredicted, it takes 5 cycles for the next instruction to reach the execute stage.

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

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

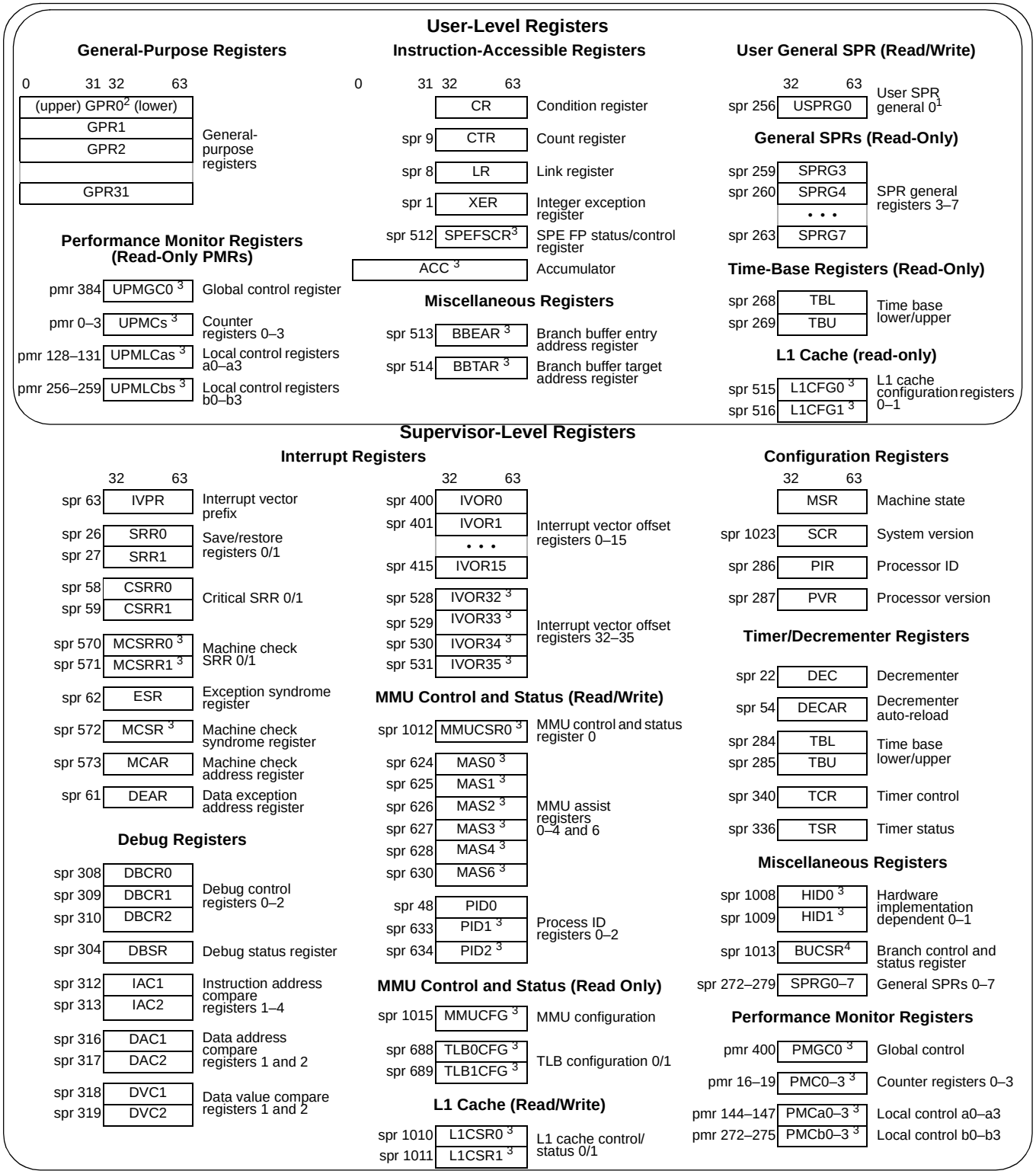
  The write-back stage occurs in the clock cycle after the instruction is retired.

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

As shown in Figure 5-4, the LSU, MU, and SU1 replicate logic to support 64-bit operations. Although a vector instruction generates separate, discrete results in the upper and lower halves of the target GPR, latency and throughput for vector instructions are the same as those for their scalar equivalents.

5.6 Programming Model

The following section describes the e500 core registers defined in Book E, the Freescale Semiconductor 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 defined by the EIS and are not part of the Book E architecture.
4 These registers are e500-specific.

Figure 5-6. e500 Core Programming Model

5.7 On-Chip Cache Implementation

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

The data cache supports four-state MESI memory coherency protocol 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.

On the MPC8560 the ABE bit must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.

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

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

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

5.8 Interrupts and Exception Handling

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

5.8.1 Exception Handling

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

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

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

5.8.2 Interrupt Classes

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

- Asynchronous interrupts (such as machine check, critical input, and external interrupts) are caused by events that are independent of instruction execution. For asynchronous interrupts, the address reported in a save/restore register is the address of the instruction that would have executed next had the asynchronous interrupt not occurred.
- Synchronous interrupts are those that are caused directly by the execution or attempted execution of instructions. Synchronous inputs may be either precise or imprecise, which are described as follows:
  - Synchronous precise interrupts are those that precisely indicate the address of the instruction causing the exception that generated the interrupt or, in some cases, the address of the immediately following instruction. The interrupt type and status bits indicate which instruction is addressed in the appropriate save/restore register.
  - Synchronous imprecise interrupts are those that may indicate the address of the 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.

5.8.3 Interrupt Types

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

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

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 noncritical interrupts can be masked by the external interrupt enable bit, MSR[EE].

3. Critical interrupts—Critical interrupts can be taken during a noncritical interrupt or during regular program flow. They use the critical save and restore registers (CSRR0/CSRR1) to save state when they are taken and they use the **rfci** instruction to restore state. These interrupts can be masked by the critical enable bit, MSR[CE]. 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 in Table 5-6.

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

5.8.4 Upper Bound on Interrupt Latencies

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

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

5.8.5 Interrupt Registers

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

Table 5-6. Interrupt Registers

Register	Description
Noncritical Interrupt Registers	
SRR0	Save/restore register 0—Holds the address of the instruction causing the exception or the address of the instruction that will execute after the **rfi** instruction.
SRR1	Save/restore register 1—Holds machine state on noncritical interrupts and restores machine state after an **rfi** instruction is executed.

Table 5-6. Interrupt Registers (continued)

Register	Description
Critical Interrupt Registers	
CSRR0	Critical save/restore register 0—On critical interrupts, holds either the address of the instruction causing the exception or the address of the instruction that will execute after the **rfci** instruction.
CSRR1	Critical save/restore register 1—Holds machine state on critical interrupts and restores machine state after an **rfci** instruction is executed.
Machine Check Interrupt Registers	
MCSRR0	Machine check save/restore register 0—Used to store the address of the instruction that will execute after an **rfmci** instruction is executed.
MCSRR1	Machine check save/restore register 1—Holds machine state on machine check interrupts and restores machine state (if recoverable) after an **rfmci** instruction is executed.
MCAR	Machine check address register—Holds the address of the data or instruction that caused the machine check interrupt. MCAR contents are not meaningful if a signal triggered the machine check interrupt.
Syndrome Registers	
MCSR	Machine check syndrome register—Holds machine state information on machine check interrupts and restores machine state after an **rfmci** instruction is executed.
ESR	Exception syndrome register—Provides a syndrome to differentiate between the different kinds of exceptions that generate the same interrupt type. Upon generation of a specific exception type, the associated bit is set and all other bits are cleared.
SPE 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.
Other Interrupt Registers	
DEAR	Data exception address register—Holds the address that was referenced by a load, store, or cache management instruction that caused an alignment, data TLB miss, or data storage interrupt.
IVPR IVORs	Together, IVPR[32–47]||IVOR*n* [48–59]||0b0000 define the address of an interrupt-processing routine. See Table 5-7 and the EREF for more information.

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

Table 5-7. Interrupt Vector Registers and Exception Conditions

Register	Interrupt
Book E–Defined IVORs	
IVOR0	Critical input
IVOR1	Machine check interrupt offset

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

Register	Interrupt
IVOR2	Data storage interrupt offset
IVOR3	Instruction storage interrupt offset
IVOR4	External input interrupt offset
IVOR5	Alignment interrupt offset
IVOR6	Program interrupt offset
IVOR7	Floating-point unavailable interrupt offset
IVOR8	System call interrupt offset
IVOR9	Auxiliary processor unavailable interrupt offset
IVOR10	Decrementer interrupt offset
IVOR11	Fixed-interval timer interrupt offset
IVOR12	Watchdog timer interrupt offset
IVOR13	Data TLB error interrupt offset
IVOR14	Instruction TLB error interrupt offset
IVOR15	Debug interrupt offset
e500-Specific IVORs	
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

5.9 Memory Management

The e500 core complex supports demand-paged virtual memory as well other memory management schemes that depend on precise control of effective-to-physical address translation and flexible memory protection as defined by 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) (ignored on the MPC8560)
- 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.

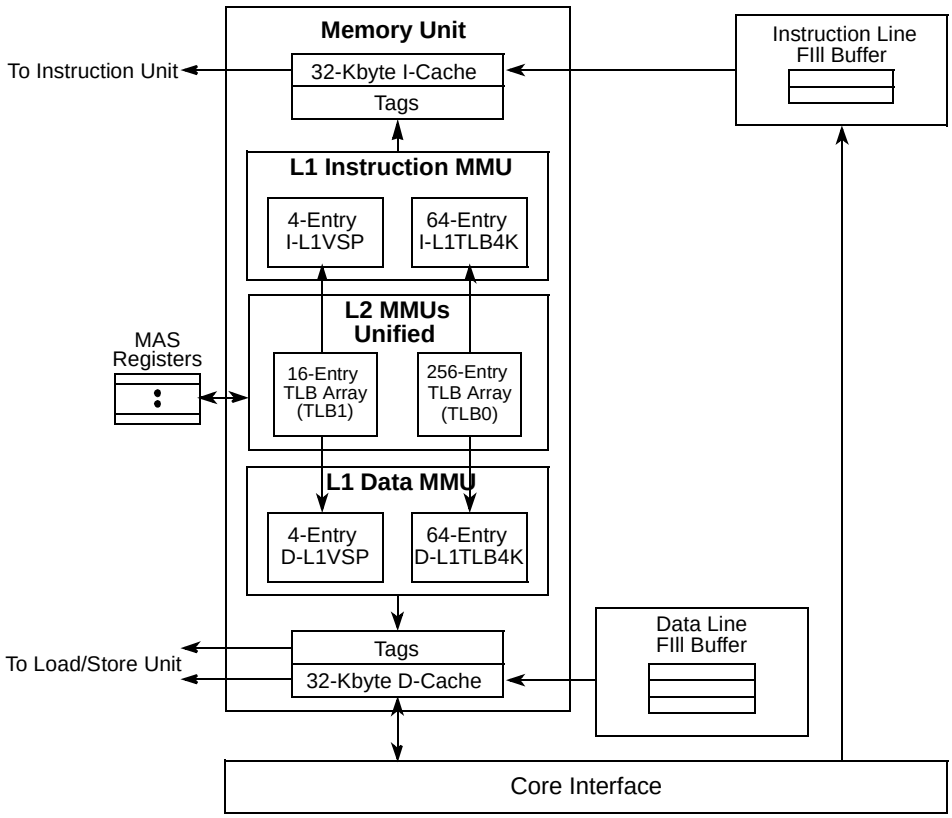


Figure 5-7. MMU Structure

Level-1 MMUs have the following features:

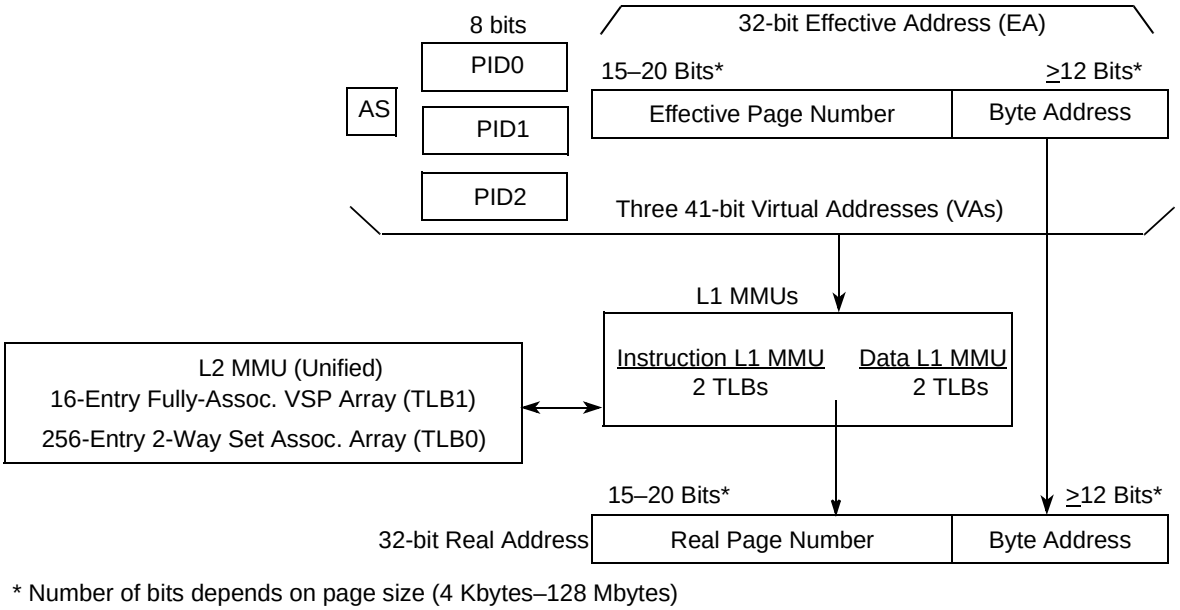
- Four-entry, fully associative TLB array that supports all nine page sizes
- 64-entry, 4-way set-associative TLB 4-Kbyte array that supports 4-Kbyte pages only
- Hardware partially managed by L2 MMU
- Supports snooping of TLBs by both internal and external **tlbivax** instructions

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

5.9.1 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. Figure 5-8 shows the translation flow.



* Number of bits depends on page size (4 Kbytes–128 Mbytes)

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.

5.9.2 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. To ensure consistency among Freescale Semiconductor Book E processors, certain aspects of the implementation are defined by the Freescale Semiconductor 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.

5.9.3 Process ID Registers (PID0–PID2)

The e500 core complex also implements three process ID (PID) registers that hold the values used to construct the three virtual addresses for each access. These process IDs provide an extended page sharing capability. Which of these three virtual addresses is used is controlled by the TID field of a matching TLB entry, and when TID = 0x00 (identifying a page as globally shared), the PID values are ignored.

A hit to multiple TLB entries in the L1 MMU (even if they are in separate arrays) or a hit to multiple entries in the L2 MMU is considered to be a programming error.

5.9.4 TLB Coherency

The core complex provides the ability to invalidate a TLB entry as defined 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] is set. The core complex also snoops TLB invalidate transactions on the CCB from other bus masters.

On the MPC8560 the ABE bit must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.

5.10 Memory Coherency

The core complex supports 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 four-state MESI protocol supports efficient large-scale real-time data sharing between multiple caching bus masters.

5.10.1 Atomic Update Memory References

The e500 core supports atomic update memory references for both aligned word forms of data using the load and reserve and store conditional instruction pair, **lwarx** and **stwcx.**. Typically, a load and reserve instruction establishes a reservation and is paired with a store conditional instruction to achieve the atomic operation. However, there are restrictions and requirements for this functionality. Because the processor revokes reservations during context switches, the programmer must reacquire the reservation after a context switch to maintain reservations across context switches.

5.10.2 Memory Access Ordering

The core complex supports weakly ordered references to memory. Thus the e500 manages the order and synchronization of instructions to ensure proper execution when memory is shared between multiple processes or programs. The cache and data memory control attributes along with **msync** and **mbar** provide the required access control.

5.10.3 Cache Control Instructions

The core complex supports 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**)

5.10.4 Programmable Page Characteristics

Cache and memory attributes are programmable on a per-page basis. In addition to the write-through, caching-inhibited, memory coherency enforce, and guarded characteristic defined by the WIMG bits, Book E defines an endianness bit, E, that allows selection of big- or little-endian byte ordering on a per-page basis.

In addition to the WIMGE bits, the Book E MMU model defines user-definable page attribute bits (U0–U3).

5.11 Core Complex Bus (CCB)

The core complex defines a versatile local bus interface that allows a wide range of system performance and system-complexity trade-offs. The interface defines the following buses.

- An address-out bus for mastering bus transactions
- An address-in bus for snooping internal resources
- Three tagged data buses

Two of the data buses are general-purpose data-in buses for reads, and the third is a data-out bus for writes. The two data-in buses feature support for out-of-order read transactions from two different sources simultaneously, and all three data buses may be operated concurrently. The address-in bus supports snooping for external management of the L1 caches and TLBs by other bus masters. The core complex broadcasts and snoops the cache and TLB management instructions accordingly. It is envisioned that a wide range of system implementations can be constructed from the defined interface.

5.12 Performance Monitoring

The e500 core provides a performance monitoring capability that allows counting of events such as processor clocks, instruction cache misses, data cache misses, mispredicted branches, and others. The count of these events may be configured to trigger a performance monitor exception following the e500 interrupt model. This interrupt is assigned to vector offset register IVOR35.

The register set associated with the performance monitoring function consists of counter registers, a global control register, and local control registers. These registers are read/write from supervisor mode, and each register is reflected to a corresponding read-only register for user mode. Two instructions, **mtpmr** and **mfpmr**, are provided for moving data to and from these registers. An overview of the performance monitoring registers is provided in the following sections.

5.12.1 Global Control Register

The PMGC0 register provides global control of the performance monitoring facility from supervisor mode. From this register all counters may be frozen, unfrozen, or configured to freeze on an enabled condition or event. Additionally, the performance monitoring facility may be disabled or enabled from this register. The contents of PMGC0 are reflected to UPMGC0 which may be read from user mode using the **mfpmr** instruction.

5.12.2 Performance Monitor Counter Registers

There are four counter registers (PCM0–PCM3) provided in the performance monitoring facility. These 32-bit registers hold the current count for software selectable events and can be programmed to generate an exception on overflow. These registers may be written or read from supervisor mode using the **mtpmr** and **mfpmr** instructions. The contents of these registers are reflected to UPCM0–UPCM3, which can be read from user mode with **mfpmr**.

Performance monitor exceptions occur only if all of the following conditions are met:

- A counter is in the overflow state
- The counter's overflow signalling is enabled
- Overflow exception generation is enabled in PMGC0
- MSR[EE] is set

5.12.3 Local Control Registers

For each of the counter registers, there are two corresponding local control registers. These two registers specify which of the 128 available events is to be counted, what specific action is to be taken on overflow, and various options for freezing a counter value under given modes or conditions.

- PMLCa0–PMLCa3 provide fields that allow freezing of the corresponding counter in user mode, supervisor mode, or under software control. Additionally, the overflow condition may be enabled or disabled from this register. The contents of these registers are reflected to 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 eight multiplier values in the range of 1 to 128. Any counter may be configured to increment only when an event occurs more than [threshold × multiplier] times. The contents of these registers are reflected to UPMCLb0–UPMLCb3, which can be read from user mode with **mfpmr**.

5.13 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.

5.13.1 Instruction Set Compatibility

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

5.13.1.1 User Instruction Set

The e500 core executes legacy user-mode binaries and object files except for the following:

- The e500 supports vector and scalar single-precision floating-point operations as APUs. These instructions have 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.
- String instructions are not implemented on the e500; therefore, trap emulation must be provided to ensure backward compatibility.

5.13.1.2 Supervisor Instruction Set

The supervisor mode instruction set defined by the 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.

5.13.2 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.

5.13.3 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 machine check exception differs 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, the fixed-interval timer critical interrupt, and the watchdog timer interrupt, which are implemented in the e500 core.

An overview of the interrupt and exception handling capabilities of the e500 core can be found in Section 5.8, “Interrupts and Exception Handling.”

5.13.4 Memory Management

The e500 core implements a straightforward virtual address space that complies with the Book E MMU definition, which eliminates 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.

5.13.5 Reset

Book E–compliant cores do not share a common reset vector with the AIM version of the PowerPC architecture. Instead, at reset fetching begins at address 0xFFFF_FFFC. In addition to

the Book E reset definition, the EIS and the e500 define specific aspects of the MMU page translation and protection mechanisms. Unlike the AIM version of the PowerPC core, as soon as instruction fetching begins, the e500 core is in virtual mode with a hardware-initialized TLB entry.

5.13.6 Little-Endian Mode

Unlike the AIM version of the PowerPC architecture, where little-endian mode is controlled on a system basis, Book E allows control of byte ordering on a memory page basis. In addition, the little-endian mode used in Book E is true little endian.

5.14 PowerQUICC III Implementation Details

Table 5-8 summarizes e500 core functionality that is not implemented by PowerQUICC III devices.

Table 5-8. Differences Between the e500 Core and the PowerQUICC III Core Implementation

Feature	PowerQUICC Implementation
Cache protocol	The L2 cache does not support MESI cache protocol.
Multiprocessor functionality	Because PowerQUICC III is designed for a uniprocessor environment, the following e500 functionality is not implemented: • The memory coherence bit, M, which is controlled through MAS2[M] and MAS4[MD] has no effect. • HID1[ABE] has meaning only in that it must be set to ensure that cache and TLB management instructions operate properly with respect to the L2 cache. • Dynamic snooping does not occur in power-stopped state (see the note below in the entry for dynamic bus snooping).
Nexus support	Nexus is not supported. The Nexus processor ID register (NPIDR) and the Nexus bus enable bit (HID1[NEXEN]) are not supported.
R1 and R2 data bus parity	R1 and R2 data bus parity are disabled on PowerQUICC III devices. HID1[R1DPE,R2DPE] are reserved.
Dynamic bus snooping	The PowerQUICC III devices do not perform dynamic bus snooping as described here. That is, when the e500 core is in core-stopped state (which is the state of the core when the PowerQUICC III devices is in either the nap or sleep state), the core is not awakened to perform snoops on global transactions. Therefore, before entering nap or sleep modes, L1 caches should be flushed if coherency is required during these power-down modes. For more information, see Section 18.5.1.9, “Snooping in Power-Down Modes.”
Supported TCR[WRC]	PowerQUICC III devices define values for 01, 10, and 11, as follows: 00 No watchdog timer reset can occur. 01 Force processor checkstop on second timeout of watchdog timer 10 Assert processor reset output (*core_hreset_req*) on second timeout of watchdog timer 11 Reserved
SPE and SPFP APUs	The SPE and SPFP APU functionality will not be implemented in next generation of PowerQUICC devices. Freescale Semiconductor strongly recommends that use of these instructions be confined to libraries and device drivers. Customer software that uses SPE or SPFP instructions at the assembly level or that uses SPE or SPFP intrinsics will require rewriting for upward compatibility with next generation PowerQUICC devices. Freescale Semiconductor offers a libmoto_e500 library that uses SPE and SPFP APU instructions. Freescale Semiconductor will also provide future libraries to support next generation PowerQUICC devices.

Table 5-8. Differences Between the e500 Core and the PowerQUICC III Core Implementation (continued)

Feature	PowerQUICC Implementation
HID0 implementation	SEL_TBCLK bit. Selects time base clock. If this bit is set and the time base is enabled, the time base is based on the TBCLK input, which on the PowerQUICC III devices is RTC.
HID1 Implementation	PLL_MODE Set to 01 PLL_CFG. PowerQUICC III devices support the following: 00010 0 2:1 00010 1 5:2 (2.5:1) 00011 0 3:1 00011 1 7:2 (3.5:1) NEXEN, R1DPE, R2DPE, MPXTT, MSHARS, SSHAR, ATS, and MID are not implemented On PowerQUICC III devices, ABE must be set to ensure that cache and TLB management instructions operate properly on the L2 cache.
	HID1[RFXE] controls whether assertion of *core_fault_in* causes a machine check interrupt. Assertion of *core_fault_in* can result from uncorrectable data error, such as an L2 multibit ECC error. It can also occur for a system error if logic on the integrated device signals a fault for nonfatal errors (read data is corrupt (or zero), but the transaction can complete without corrupting other system state, making it unnecessary to take a machine check (for example, a master abort of a PCI transaction). If RFXE is 0, and *core_fault_in* is asserted, any data on the CCB is dropped, either stalling the load/store unit and causing the e500 pipeline to stall until an interrupt occurs (typically generated by the programmable interrupt controller (PIC) in response to the fault) or allowing processingþto continue with the bad data until the interrupt occurs. Because *core_fault_in* cannot cause a machine check if RFXE is 0, it is critical that the system be configured to generate the appropriate interrupt, as described below. It is also possible to hang the processor (requiring a hard reset to recover) if a guardedþload hits in the L2 cache and gets an uncorrectable ECC error. Because of this, avoid defining memory as cacheable but guarded. If this combination is required, RFXE must be enabled, in which case an error causes both a machine check interrupt and an external interrupt when a bus fault condition is detected unless interrupts are masked for all sources of bus faults. If RFXE is 0, conditions that cause the assertion of core_fault_in cannot directly cause the e500 to generate a machine check; however, PowerQUICC III devices must be configured to detect and enable such conditions. The following describes how error bits should be configured: • ECM mapping errors: EEER[LAEE] must be set. See Section 8.2.1.4, “ECM Error Enable Register (EEER).” • L2 multiple-bit ECC errors: L2ERRDIS[MBECCDIS] must be cleared to ensure that error can be detected. L2ERRINTEN[MBECCINTEN] must be set. SeeSection 7.3.1.5, “L2 Error Registers.” • DDR multiple-bit ECC errors. ERR_DISABLE[MBED] and ERR_INT_EN[MBEE] must be zero and DDR_SDRAM_CFG[ECC_EN] must be one to ensure that an interrupt is generated. See Section 9.4.1, “Register Descriptions.” • PCI. The appropriate parity detect and master-abort bits in ERR_DR must be cleared and the corresponding enable bits in ERR_EN must be set to ensure that an interrupt is generated. Section 15.3.1.4, “PCI/X Error Management Registers.” • Local bus controller parity errors. LTEDR[PARD] must be cleared and LTEIR[PARI] must be set to ensure that an parity errors can generate an interrupt. See Section 12.3.1.11, “Transfer Error Check Disable Register (LTEDR),” and Section 12.3.1.12, “Transfer Error Interrupt Enable Register (LTEIR).” • • RapidIO. PCR[CCE] must be set to ensure that an interrupt is generated due to a CRC error. See Section 16.3.2.1.2, “Port Configuration Register (PCR).” RFXE must also be set if software requires that code execution stop immediately when a bus fault occurs rather than continuing with the bad data until the interrupt arrives. Again, this results in both a machine check interrupt and an external interrupt when a bus fault is detected, unless all possible sources for bus fault have their interrupts masked.þThe machine check interrupt can then reenable normal interrupts and wait for the interrupt due to the fault to be received before returning from the machine check.

Table 5-8. Differences Between the e500 Core and the PowerQUICC III Core Implementation (continued)

Feature	PowerQUICC Implementation
PIR value	The PIR value is all zeros on PowerQUICC III devices.
PVR value	The PVR reset value is 0x8020_*nnnn.* See Table 5-1. PVR[VERSION] = 0x8020 PVR[REVISION] = 0x*nnnn* (revision specific value)
SVR value	The SVR reset value is. See Table 5-1. 0x8070_*nnnn* SVR[VERSION] = 0x8070 SVR[REVISION] = 0x*nnnn* (revision specific value)

Chapter 6
Core Register Summary

This chapter describes the e500 register model and indicates whether each register is defined by Book E, by the Freescale Semiconductor Book E implementation standards (EIS), or by the implementation. For the programmer, drawing this distinction indicates the degree to which code is portable among Freescale Semiconductor Book E processors.

This chapter provides reference material—figures for each register and complete descriptions of register fields, including how the registers are accessed, reset values, and whether they can be accessed by user- and supervisor-level software. Detailed discussions of how these registers are used are provided in individual chapters.

Note that all registers described here are implemented in the hardware as part of the e500 core.

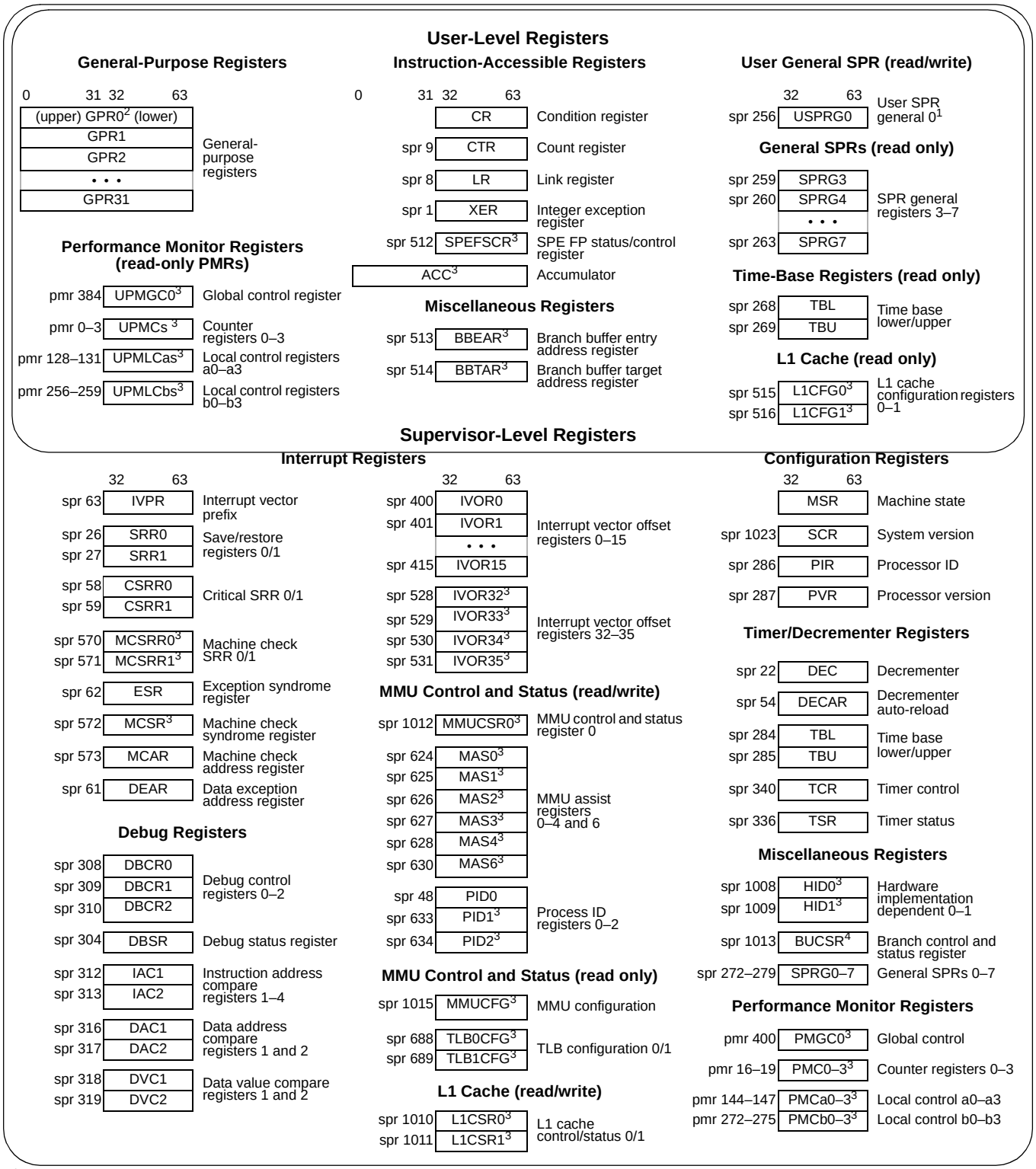
6.1 Overview

As shown in Figure 6-1, most of the registers implemented on a Freescale Semiconductor Book E processor are defined by the Book E architecture, and most of those were defined by the AIM definition of the PowerPC architecture and have changed very little. Additional registers and fields within Book E–defined registers are defined by the EIS and by the implementation.

Book E defines some register fields in a very general way, leaving some details as implementation specific. In some cases, this more specific functionality is defined by the EIS; in others it is left up to the processor. This chapter identifies the level at which each features is defined.

6.1.1 Register Set

Table 6-1 shows the e500 register set, grouped by whether they can be accessed by user- or supervisor-level software. Unless otherwise indicated, these registers are defined by Book E.



1 USPRG0 is a separate physical register from SPRG0.
2 The 64-bit GPR registers are accessed by the SPE as separate 32-bit registers by SPE instructions. Only SPE vector instructions can access the upper word.
3 These registers are defined by the EIS and are not part of the Book E architecture.
4 These registers are e500-specific.

Figure 6-1. Core Register Model

6.2 Register Model for 32-Bit Implementations

Embedded 32-bit processors implement the following types of software-accessible registers:

- Book E–defined registers that are accessed as part of instruction execution. These include the following:
  - Registers used for computation. These include the following:
    - General-purpose registers (GPRs)—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. The e500 implements these as 64-bit registers for use with 64-bit load, store, and merge instructions, as described in Section 6.3.1, “General-Purpose Registers (GPRs).”
    - Integer exception register (XER)—Bits in this register are set based on the operation of an instruction considered as a whole, not on intermediate results. (For example, the Subtract from Carrying instruction (**subfc**), the result of which is specified as the sum of three values, sets bits in the XER based on the entire operation, not on an intermediate sum.)

    These registers are described in Section 6.3, “Registers for Computational Operations.”
  - Condition register (CR)—Used to record conditions such as overflows and carries that occur as a result of executing arithmetic instructions (including those implemented by the SPE 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 **mtspr** and **mfspr** instructions. These registers are listed in Table 6-1 in Section 6.2.1, “Special-Purpose Registers (SPRs).”
- Freescale Semiconductor EIS– and e500-defined SPRs that are accessed explicitly using **mtspr** and **mfspr** are listed in Table 6-2 in Section 6.2.1, “Special-Purpose Registers (SPRs).”
- Freescale Semiconductor EIS–defined performance monitor registers (PMRs). These registers are similar to SPRs, but are accessed with Freescale Semiconductor EIS–defined move to and move from PMR instructions (**mtpmr** and **mfpmr**).

In this chapter, SPRs are grouped by function as follows:

- Section 6.4, “Registers for Branch Operations,” describes the count register (CTR) and the link register (LR) defined by Book E.
- Section 6.5, “Processor Control Registers”
- Section 6.6, “Timer Registers”

- Section 6.7, “Interrupt Registers”
- Section 6.8, “Software-Use SPRs (SPRG0–SPRG7 and USPRG0),” describes Book E–defined SPRs defined for software use.
- Section 6.9, “Branch Target Buffer (BTB) Registers,” describes e500-specific registers defined to support the e500 BTBs.
- Section 6.10, “Hardware Implementation-Dependent Registers,” describes HID0 and HID1.
- Section 6.11, “L1 Cache Configuration Registers”
- Section 6.12, “MMU Registers”
- Section 6.13, “Debug Registers”
- Section 6.14, “Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)”

The e500 core implements 64-bit GPRs, the upper 32 bits of which are used only with 64-bit load, store, and merge instructions.

6.2.1 Special-Purpose Registers (SPRs)

Table 6-1 summarizes SPRs defined in Book E. The SPR numbers are used in the instruction mnemonics. Bit 5 in an SPR number indicates whether an SPR is accessible from user- or supervisor-level software. An **mtspr** or **mfspr** instruction that specifies an unsupported SPR number is considered an invalid instruction.

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

SPR Abbreviation	Name	Defined SPR Number		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
CSRR0	Critical save/restore register 0	58	00001 11010	Read/Write	Yes	6.7.1.1/6-17
CSRR1	Critical save/restore register 1	59	00001 11011	Read/Write	Yes	6.7.1.2/6-17
CTR	Count register	9	00000 01001	Read/Write	No	6.4.3/6-11
DAC1	Data address compare 1	316	01001 11100	Read/Write	Yes	6.13.4/6-45
DAC2	Data address compare 2	317	01001 11101			
DBCR0	Debug control register 0	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	6.13.2/6-43
DEAR	Data exception address register	61	00001 11101	Read/Write	Yes	6.7.1.5/6-18
DEC	Decrementer	22	00000 10110	Read/Write	Yes	6.6.4/6-16
DECAR	Decrementer auto-reload	54	00001 10110	Write-only		
ESR	Exception syndrome register	62	00001 11110	Read/Write	Yes	6.7.1.8/6-19

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

SPR Abbreviation	Name	Defined SPR Number		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
IAC1	Instruction address compare 1	312	01001 11000	Read/Write	Yes	6.13.3/6-44
IAC2	Instruction address compare 2	313	01001 11001			
IVOR0	Critical input	400	01100 10000	Read/Write	Yes	6.7.1.7/6-18
IVOR1	Machine check interrupt offset	401	01100 10001			
IVOR2	Data storage interrupt offset	402	01100 10010			
IVOR3	Instruction storage interrupt offset	403	01100 10011			
IVOR4	External input interrupt offset	404	01100 10100			
IVOR5	Alignment interrupt offset	405	01100 10101			
IVOR6	Program interrupt offset	406	01100 10110			
IVOR8	System call interrupt offset	408	01100 11000			
IVOR10	Decrementer interrupt offset	410	01100 11010			
IVOR11	Fixed-interval timer interrupt offset	411	01100 11011			
IVOR12	Watchdog timer interrupt offset	412	01100 11100			
IVOR13	Data TLB error interrupt offset	413	01100 11101			
IVOR14	Instruction TLB error interrupt offset	414	01100 11110			
IVOR15	Debug interrupt offset	415	01100 11111			
IVPR	Interrupt vector	63	00001 11111	Read/Write	Yes	6.7.1.6/6-18
LR	Link register	8	00000 01000	Read/Write	No	6.4.2/6-10
PID	Process ID register [2]	48	00001 10000	Read/Write	Yes	6.12.1/6-32
PIR	Processor ID register	286	01000 11110	Read-only	Yes	6.5.2/6-13
PVR	Processor version register	287	01000 11111	Read-only	Yes	6.5.3/6-13
SPRG0	SPR general 0	272	01000 10000	Read/Write	Yes	6.8/6-23
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	6.8/6-23
		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	

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

SPR Abbreviation	Name	Defined SPR Number		Access	Supervisor Only	Section/ Page
		Decimal	Binary			
SRR0	Save/restore register 0	26	00000 11010	Read/Write	Yes	6.7.1.1/6-17
SRR1	Save/restore register 1	27	00000 11011	Read/Write	Yes	6.7.1.2/6-17
TBL	Time base lower	268	01000 01100	Read-only	No	6.6.3/6-16
		284	01000 11100	Write-only	Yes	
TBU	Time base upper	269	01000 01101	Read-only	No	
		285	01000 11101	Write-only	Yes	
TCR	Timer control register	340	01010 10100	Read/Write	Yes	6.6.1/6-14
TSR	Timer status register	336	01010 10000	Read/Clear[4]	Yes	6.6.2/6-15
USPRG0	User SPR general 0[5]	256	01000 00000	Read/Write	No	6.8/6-23
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. For implementations with multiple PIDs, the EIS 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 describes the implementation-specific SPRs. Compilers should recognize the mnemonic names given in Table 6-2 when parsing instructions.

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

SPR Abbreviation	Name	SPR Number	Access	Supervisor Only	Section/ Page
BBEAR	Branch buffer entry address register	513	Read/Write	No	6.9.1/6-23
BBTAR	Branch buffer target address register	514	Read/Write	No	6.9.2/6-24
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	6.10.1/6-25
HID1	Hardware implementation dependent reg 1	1009	Read/Write	Yes	6.10.1/6-25
IVOR32	SPE APU unavailable interrupt offset	528	Read/Write	Yes	6.7.1.7/6-18
IVOR33	Floating-point data exception interrupt offset	529	Read/Write	Yes	
IVOR34	Floating-point round exception interrupt offset	530	Read/Write	Yes	
IVOR35	Performance monitor	531	Read/Write	Yes	
L1CFG0	L1 cache configuration register 0	515	Read-only	No	6.11.3/6-30

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

SPR Abbreviation	Name	SPR Number	Access	Supervisor Only	Section/ Page
L1CFG1	L1 cache configuration register 1	516	Read-only	No	6.11.4/6-31
L1CSR0	L1 cache control and status register 0	1010	Read/Write	Yes	6.11.1/6-28
L1CSR1	L1 cache control and status register 1	1011	Read/Write	Yes	6.11.2/6-29
MAS0	MMU assist register 0	624	Read/Write	Yes	6.12.5.1/6-35
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
MCAR	Machine check address register	573	Read-only	Yes	6.7.2.3/6-21
MCSR	Machine check syndrome register	572	Read/Write	Yes	6.7.2.4/6-21
MCSRR0	Machine check save/restore register 0	570	Read/Write	Yes	6.7.2.1/6-20
MCSRR1	Machine check save/restore register 1	571	Read/Write	Yes	6.7.2.2/6-21
MMUCFG	MMU configuration register	1015	Read-only	Yes	6.12.3/6-32
MMUCSR0	MMU control and status register 0	1012	Read/Write	Yes	6.12.2/6-32
PID0	Process ID register 0. Book E refers to this as PID instead of PID0.	48	Read/Write	Yes	6.12.1/6-32
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	6.14/6-45
SVR	System version register	1023	Read-only	Yes	6.5.4/6-14
TLB0CFG	TLB configuration register 0	688	Read-only	Yes	6.12.4/6-33
TLB1CFG	TLB configuration register 1	689	Read-only	Yes	6.12.4.2/6-34

6.3 Registers for Computational Operations

The following sections describe general-purpose and integer exception registers.

NOTE

Register fields designated as write-one-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

6.3.1 General-Purpose Registers (GPRs)

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, although only 64-bit load, store, and merge instructions use GPR bits 0–31.

6.3.2 Integer Exception Register (XER)

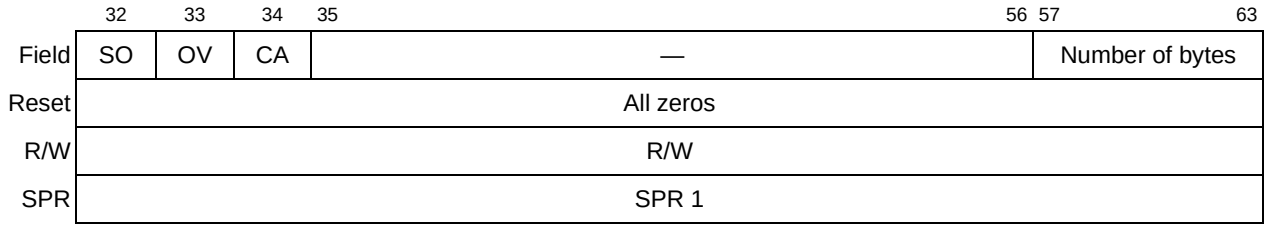


Figure 6-2. Integer Exception Register (XER)

Table 6-3. XER Field Description

Bits	Name	Description
32	SO	Summary overflow. Set when an instruction (except **mtspr**) sets the overflow bit. Once set, SO remains set until it is cleared by **mtspr[XER]** or **mcrxr**. SO is not altered by compare instructions or by other instructions (except **mtspr[XER]** and **mcrxr**) that cannot overflow. Executing **mtspr[XER]**, supplying the values 0 for SO and 1 for OV, causes SO to be cleared and OV to be set.
33	OV	Overflow. X-form add, subtract from, and negate instructions having OE = 1 set OV if the carry out of bit 32 is not equal to the carry out of bit 33, and clear OV otherwise to indicate a signed overflow. X-form multiply low word and divide word instructions having OE = 1 set OV if the result cannot be represented in 32 bits (**mullwo**, **divwo**, and **divwuo**) and clear OV otherwise. OV is not altered by compare instructions or by other instructions (except **mtspr[XER]** and **mcrxr**) that cannot overflow.
34	CA	Carry. Add carrying, subtract from carrying, add extended, and subtract from extended instructions set CA if there is a carry out of bit 32 and clear it otherwise. CA can be used to indicate unsigned overflow for add and subtract operations that set CA. Shift right algebraic word instructions set CA if any 1 bits are shifted out of a negative operand and clear CA otherwise. Compare instructions and instructions that cannot carry (except Shift Right Algebraic Word, **mtspr[XER]**, and **mcrxr**) do not affect CA.
35–56	—	Reserved, should be cleared.
57–63	No. of bytes	Supports emulation of load and store string instructions. Specifies the number of bytes to be transferred by a load string indexed or store string indexed instruction.

6.4 Registers for Branch Operations

This section describes registers that support Book E branch and CR operations.

6.4.1 Condition Register (CR)

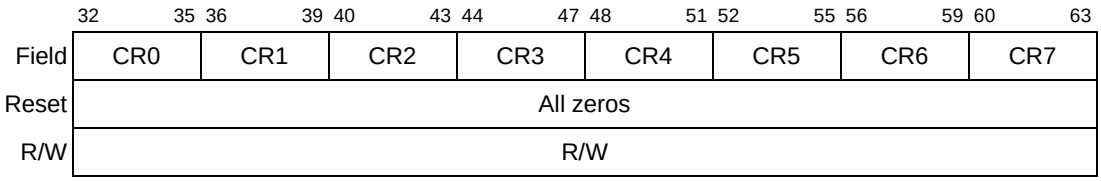


Figure 6-3. Condition Register (CR)

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	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	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	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	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.
CR*n*[0]	40 44 48 52 56 60	01000 01100 10000 10100 11000 11100	Less than (LT) For integer compare instructions: **r**A < SIMM or **r**B (signed comparison) or **r**A < UIMM or **r**B (unsigned comparison). 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.

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

CR*n* Bits	CR Bits	BI	Description
CR*n*[1]	41 45 49 53 57 61	01001 01101 10001 10101 11001 11101	Greater than (GT) For integer compare instructions: **r**A > SIMM or **r**B (signed comparison) or **r**A > UIMM or **r**B (unsigned comparison). For SPE 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.
CR*n*[2]	42 46 50 54 58 62	01010 01110 10010 10110 11010 11110	Equal (EQ) For integer compare instructions: **r**A = SIMM, UIMM, or **r**B. For SPE 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 (SO) For integer compare instructions, this is a copy of XER[SO] at the completion of the instruction. 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.

The bits of CR0 are interpreted as described in Table 6-5.

Table 6-5. CR0 Bit Descriptions

CR Bit	Name	Description
32	Negative (LT)	Bit 32 of the result is equal to 1.
33	Positive (GT)	Bit 32 of the result is equal to 0 and at least one bit from 33–63 of the result is non-zero.
34	Zero (EQ)	Bits 32–63 of the result are equal to 0.
35	Summary overflow (SO)	This is a copy of the final state of XER[SO] at the completion of the instruction.

6.4.2 Link Register (LR)

	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

This section addresses machine state, processor ID, and processor version registers.

6.5.1 Machine State Register (MSR)

	32–36	37	38	39–44	45	46	47	48	49	50	51	52	53	54	55–57	58	59	60	61	62–63
Field	—	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 6-6. MSR Field Descriptions

Bits	Name	Description
32–36	—	Reserved, should be cleared.[1]
37	UCLE	User-mode cache lock enable (defined by the EIS). 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 (defined by the EIS) 0 If software attempts to execute an SPE APU or SPFP instruction, an SPE APU unavailable exception is taken. 1 Software can execute supported SPE and SPFP APU instructions. **Note:** The SPE APU and SPFP APU functionality will be implemented in the all PowerQUICC III devices. However, these instructions will not be supported in devices subsequent to PowerQUICC III. Freescale Semiconductor strongly recommends that use of these instructions be confined to libraries and device drivers. Customer software that uses SPE or SPFP APU instructions at the assembly level or that uses SPE intrinsics will require rewriting for upward compatibility with next generation PowerQUICC devices. Freescale Semiconductor offers a libmoto_e500 library that uses SPE and SPFP APU instructions. Freescale will also provide future libraries to support next-generation PowerQUICC devices.

Table 6-6. MSR Field Descriptions (continued)

Bits	Name	Description
39–44	—	Reserved, should be cleared. [1]
45	WE	Wait state enable. Allows the core complex to signal a request for power management, according to the states of HID0[DOZE], HID0[NAP], and HID0[SLEEP]. 0 The processor is not in wait state and continues processing. No power management request is signaled to external logic. 1 The processor enters wait state by ceasing to execute instructions and entering low-power mode. Details of how wait state is entered and exited and how the processor behaves in the wait state are implementation-dependent. On the e500, MSR[WE] gates the DOZE, NAP, and SLEEP outputs from the core complex; as a result, these outputs negate to the external power management logic on entry to the interrupt and then return to their previous state on return from the interrupt. WE is cleared on entry to any interrupt and restored to its previous state upon return.
46	CE	Critical enable 0 Critical input and watchdog timer interrupts are disabled. 1 Critical input and watchdog timer interrupts are enabled.
47	—	Preserved for Book III ILE
48	EE	External enable 0 External input, decrementer, fixed-interval timer, and performance monitor interrupts are disabled. 1 External input, decrementer, fixed-interval timer, and performance monitor interrupts are enabled.
49	PR	User mode (problem state) 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (for example, GPRs, SPRs, and the MSR). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource. PR also affects memory access control
50	—	Reserved, should be cleared. [1]
51	ME	Machine check enable 0 Machine check interrupts are disabled. 1 Machine check interrupts are enabled.
52	—	Reserved, should be cleared. [1]
53	UBLE	In the e500, it is the user BTB lock enable bit. 0 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).”
55–57	—	Reserved, should be cleared. [1]
58	IS	Instruction address space 0 The processor directs all instruction fetches to address space 0 (TS = 0 in the relevant TLB entry). 1 The processor directs all instruction fetches to address space 1 (TS = 1 in the relevant TLB entry).
59	DS	Data address space 0 The processor directs data memory accesses to address space 0 (TS = 0 in the relevant TLB entry). 1 The processor directs data memory accesses to address space 1 (TS = 1 in the relevant TLB entry).

Table 6-6. MSR Field Descriptions (continued)

Bits	Name	Description
60	—	Reserved, should be cleared. [1]
61	PMM	Performance monitor mark bit (defined by the EIS). System software can set PMM when a marked process is running to enable statistics to be gathered only during execution of the marked process. MSR[PR] and MSR[PMM] together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCax, the state for which monitoring is enabled, counting is enabled.
62–63	—	Preserved for OEA-defined RI and LE, respectively

[1] An MSR bit that is reserved may be altered by a return from interrupt instruction.

6.5.2 Processor ID Register (PIR)

	32 — 63
Field	Processor ID
Reset	All zeros
R/W	Read only
SPR	SPR 286

Figure 6-7. Processor ID Register (PIR)

6.5.3 Processor Version Register (PVR)

	32 — 47	48 — 63
Field	Version	Revision
Reset	0x8020_*nnnn*	
R/W	Read only	
SPR	SPR 287	

Figure 6-8. Processor Version Register (PVR)

Table 6-7. PVR Field Descriptions

Bits	Name	Description
32–47	Version	A 16-bit number that identifies the version of the processor. Different version numbers indicate major differences between processors, such as which optional facilities and instructions are supported.
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.

6.5.4 System Version Register (SVR)

	32	63
Field	System version	
Reset	SoC-dependent value: 0x8070_*nnnn*0x	
R/W	Read only	
SPR	SPR 1023	

Figure 6-9. System Version Register (SVR)

6.6 Timer Registers

6.6.1 Timer Control Register (TCR)

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

Figure 6-10. Timer Control Register (TCR)

Table 6-8. TCR Field Descriptions

Bits	Name	Description
32–33	WP	Watchdog timer period. When concatenated with WPEXT, specifies one of 64-bit locations of the time base used to signal a watchdog timer exception on a transition from 0 to 1. WPEXT[0–3] || WP[0–1] = 0b00_0000 selects TBU[32] (the msb of the TB) WPEXT[0–3] || WP[0–1] = 0b11_1111 selects TBL[63] (the lsb of the TB)
34–35	WRC	Watchdog timer reset control. This value is written into TSR[WRS] when a watchdog event occurs. WRC may be set by software but cannot be cleared by software, except by a software-induced reset. Once written to a non-zero value, WRC may no longer be altered by software. 00 No watchdog timer reset will occur. 01 A second timeout is ignored, regardless of the value of MSR[ME]. 10 Assert processor reset output (*core_hreset_req*) 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

Table 6-8. TCR Field Descriptions (continued)

Bits	Name	Description
38–39	FP	Fixed interval timer period. When concatenated with FPEXT, FP specifies one of 64 bit locations of the time base used to signal a fixed-interval timer exception on a transition from 0 to 1. FPEXT[0–3] || FP[0–1] = 0b00_0000 selects TBU[32] (the msb of the TB) FPEXT[0–3] || FP[0–1] = 0b11_1111 selects TBL[63] (the lsb of the TB)
40	FIE	Fixed interval interrupt enable 0 Fixed interval interrupts disabled 1 Fixed interval interrupts enabled
41	ARE	Auto-reload enable. Controls whether the DECAR value is reloaded into the DEC when the DEC value reaches 0000_0001. See *EREF: A Reference for Freescale Semiconductor Book E and the e500 Core*. 0 Auto-reload disabled 1 Auto-reload enabled
42	—	Reserved, should be cleared.
43–46	WPEXT	Watchdog timer period extension (see the description for WP)
47–50	FPEXT	Fixed-interval timer period extension (see the description for FP)
51–63	—	Reserved, should be cleared.

6.6.2 Timer Status Register (TSR)

	32	33	34 35	36	37	38 63
Field	ENW	WIS	WRS	DIS	FIS	—
Reset	All zeros					
R/W	Set by hardware. Read with **mfspr** and cleared with **mtspr** by writing ones to any TSR bit positions to be cleared and zeros in all other bit positions.					
SPR	SPR 336					

Figure 6-11. Timer Status Register (TSR)

Table 6-9. TSR Field Descriptions

Bits	Name	Description
32	ENW	Enable next watchdog time. Functions as write-one-to-clear. 0 Action on next watchdog timer time-out is to set TSR[ENW] 1 Action on next watchdog timer time-out is governed by TSR[WIS] When a watchdog timer time-out occurs while WIS = 0 and the next watchdog time-out is enabled (ENW = 1), a watchdog timer exception is generated and logged by setting WIS. This is referred to as a watchdog timer first time out. A watchdog timer interrupt occurs if enabled by TCR[WIE] and MSR[CE]. To avoid another watchdog timer interrupt once MSR[CE] is reenabled, (assuming TCR[WIE] is not cleared instead), the interrupt handler must reset TSR[WIS].
33	WIS	Watchdog timer interrupt status. Functions as write-one-to-clear. 0 A watchdog timer event has not occurred. 1 A watchdog timer event occurred. When MSR[CE] = 1 and TCR[WIE] = 1, a watchdog timer interrupt is taken. See the description of ENW for more information about how WIS is used.

Table 6-9. TSR Field Descriptions (continued)

Bits	Name	Description
34–35	WRS	Watchdog timer reset status. Functions as write-one-to-clear. Defined at reset (value = 00). Set to TCR[WRC] when a reset is caused by the watchdog timer.
36	DIS	Decrementer interrupt status. Functions as write-one-to-clear. 0 A decrementer event has not occurred. 1 A decrementer event occurred. When MSR[EE] = TCR[DIE] = 1, a decrementer interrupt is taken.
37	FIS	Fixed-interval timer interrupt status. Functions as write-one-to-clear. 0 A fixed-interval timer event has not occurred. 1 A fixed-interval timer event occurred. When MSR[EE] = 1 and TCR[FIE] = 1, a fixed-interval timer interrupt is taken.
38–63	—	Reserved, should be cleared.

6.6.3 Time Base Registers

	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 6-12. Time 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 6-13. 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 6-14. 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 6-15. 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 6-16. Save/Restore Register 1 (SRR1)

6.7.1.3 Critical Save/Restore Register 0 (CSRR0)

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

Figure 6-17. Critical Save/Restore Register 0 (CSRR0)

6.7.1.4 Critical Save/Restore Register 1 (CSRR1)

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

Figure 6-18. Critical Save/Restore Register 1 (CSRR1)

6.7.1.5 Data Exception Address Register (DEAR)

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

Figure 6-19. Data Exception Address Register (DEAR)

6.7.1.6 Interrupt Vector Prefix Register (IVPR)

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

Figure 6-20. Interrupt Vector Prefix Register (IVPR)

6.7.1.7 Interrupt Vector Offset Registers (IVOR*n*)

	32	47	48	59	60	63
Field	—		Interrupt vector offset		—	
Reset	All zeros					
R/W	R/W					
SPR	(See Table 6-10.)					

Figure 6-21. Interrupt Vector Offset Registers (IVOR*n*)

Table 6-10. 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
IVOR8	408	System call
IVOR10	410	Decrementer
IVOR11	411	Fixed-interval timer interrupt
IVOR12	412	Watchdog timer interrupt
IVOR13	413	Data TLB error
IVOR14	414	Instruction TLB error
IVOR15	415	Debug
IVOR16–IVOR31	—	Reserved for future architectural use
IVOR32	528	SPE APU unavailable (EIS defined)
IVOR33	529	Floating-point data exception (EIS defined)
IVOR34	530	Floating-point round exception (EIS defined)
IVOR35	531	Performance monitor (EIS defined)
IVOR36–IVOR63	—	Allocated for implementation-dependent use

6.7.1.8 Exception Syndrome Register (ESR)

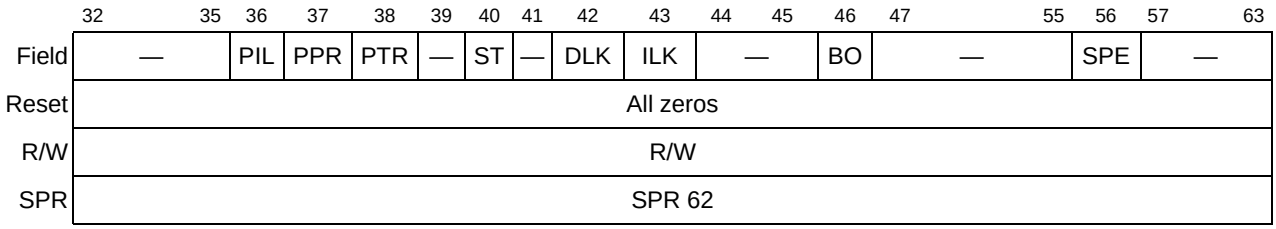


Figure 6-22. Exception Syndrome Register (ESR)

Table 6-11. ESR Field Descriptions

Bits	Name	Syndrome	Interrupt Types
32–35	—	Reserved, should be cleared. (Defined by Book E as allocated.)	—
36	PIL	Illegal instruction exception	Program
37	PPR	Privileged instruction exception	Program
38	PTR	Trap exception	Program
39	—	Defined by Book E for floating-point operations. Reserved and permanently cleared because the e500 does not implement a Book E FPU. Setting it has no effect.	—
40	ST	Store operation	Alignment, data storage, data TLB error
41	—	Reserved, should be cleared.	—
42	DLK	Cache locking. Settings are implementation-dependent. 0 Default 1 On the e500, DLK is set when a DSI occurs because **dcbtls**, **dcbtstls**, or **dcblc** is executed in user mode while MSR[UCLE] = 0.	Data storage
43	ILK	Set when a DSI occurs because **icbtl** or **icblc** is executed in user mode (MSR[PR] = 1) and MSR[UCLE] = 0	Data storage
44–45	—	Reserved, should be cleared.	
46	BO	Byte-ordering exception	Data storage, instruction storage
47–55	—	Reserved, should be cleared.	—
56	SPE	SPE exception bit (e500-specific) 0 Default 1 Any exception caused by an SPE or SPFP instruction	SPE unavailable
57–63	—	Reserved, should be cleared. (Defined by Book E as allocated.)	—

6.7.2 EIS-Defined 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 6-23. 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 6-24. Machine Check Save/Restore Register 1 (MCSRR1)

6.7.2.3 Machine Check Address Register (MCAR)

	32	63
Field	Machine check address	
Reset	All zeros	
R/W	Read only	
SPR	SPR 573	

Figure 6-25. Machine Check Address Register (MCAR)

6.7.2.4 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	—	—	—	—	—	—	—	GL_CI
Reset	All zeros							
R/W	R/W							

	48	49	50	51	52	53	54	55
Field	—	—	—	—	—	—	—	—
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 6-26. Machine Check Syndrome Register (MCSR)

Table 6-12. MCSR Field Descriptions

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.)
36–46	—	Reserved, should be cleared.	—
47	GL_CI	Set when a guarded load or cache-inhibited **stwcx.** causes a cache parity error (ICPERR, DCPERR)—MCSRR0 holds the address for the instruction after the instruction that caused the error.	
48–55	—	Reserved, should be cleared.	—
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/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 6-27. Software-Use SPRs (SPRG0–SPRG7 and USPRG0)

6.9 Branch Target Buffer (BTB) Registers

6.9.1 Branch Buffer Entry Address Register (BBEAR)

	32	61	62	63
Field	Branch buffer entry address			IAB[0–1]
Reset	All zeros			
R/W	R/W			
SPR	SPR 513			

Figure 6-28. Branch Buffer Entry Address Register (BBEAR)

Table 6-13. BBEAR Field Descriptions

Bits	Name	Description
32–61	Branch buffer entry address	Branch buffer entry effective address bits 0–29
62–63	IAB[0–1]	Instruction after branch (with BBTAR[62]). 3-bit pointer that points to the instruction in the cache line after the branch. See the description in the **bblels** instruction in the EREF. If the branch is the last instruction in the cache block, IAB = 000, to indicate the next sequential instruction, which resides in the zeroth position of the next cache block.

6.9.2 Branch Buffer Target Address Register (BBTAR)

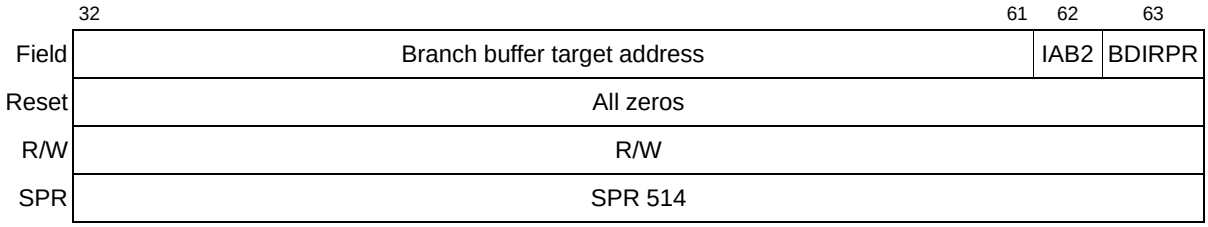


Figure 6-29. Branch Buffer Target Address Register (BBTAR)

Table 6-14. BBTAR Field Descriptions

Bits	Name	Description
32–61	Branch buffer target address	Branch buffer target effective address bits 0–29
62	IAB2	Instruction after branch bit 2 (with BBEAR[62–63]). IAB is a 3-bit pointer that points to the instruction in the cache line after the branch. See the description for **bblels** in the EREF. If the branch is the last instruction in the cache block, IAB = 000, to indicate the next sequential instruction, which resides in the zeroth position of the next cache block.
63	BDIRPR	Branch direction prediction. The user can pick the direction of the predicted branch. 0 The locked address is always predicted as not taken. 1 The locked address is always predicted as taken.

6.9.3 Branch Unit Control and Status Register (BUCSR)

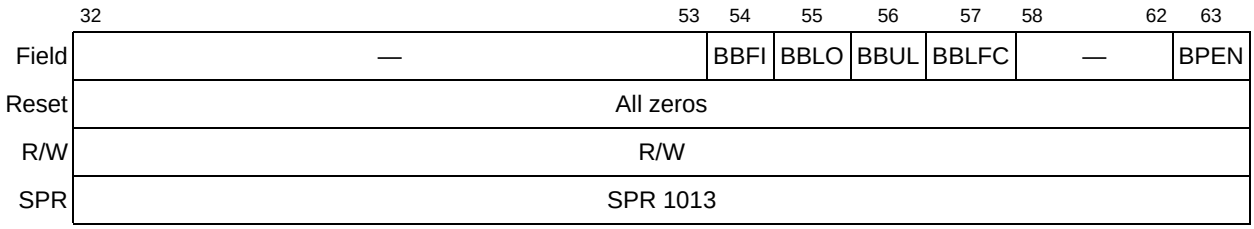


Figure 6-30. Branch Unit Control and Status Register (BUCSR)

Table 6-15. 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.

Table 6-15. BUCSR Field Descriptions (continued)

Bits	Name	Description
56	BBUL	Branch buffer unable to lock 0 Indicates a lock overflow condition in the branch buffer 1 Indicates a lock set instruction failed in the branch buffer, for example, if the BTB is disabled. This sticky bit is set by hardware and is cleared by writing 0 to this bit location.
57	BBLFC	Branch buffer lock bits flash clear. Clearing and then setting BBLFC flash clears the lock bit of all entries in the branch buffer; clearing occurs independently from the value of the enable bit (BPEN). BBLFC is always read as 0.
58–62	—	Reserved, should be cleared.
63	BPEN	Branch prediction enable 0 Branch prediction disabled 1 Branch prediction enabled (enables BTB to predict branches)

6.10 Hardware Implementation-Dependent Registers

6.10.1 Hardware Implementation-Dependent Register 0 (HID0)

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

Figure 6-31. Hardware Implementation-Dependent Register 0 (HID0)

Table 6-16. 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. Controls SLEEP mode if MSR[WE] is set. 0 Core not in sleep mode 1 Core in sleep mode

Table 6-16. HID0 Field Descriptions (continued)

Bits	Name	Description
43–48	—	Reserved, should be cleared.
49	TBEN	Time base enable 0 Time base disabled (no counting) 1 Time base enabled • If HID0[TBEN] = 1 and HID0[SEL_TBCLK] = 0, the time base is updated every 8 bus clocks • If HID0[TBEN] = 1 and HID0[SEL_TBCLK] = 1, the time base is updated on the rising edge of core_TBCLK (sampled at bus rate). The maximum supported frequency can be found in the electrical specifications, but this value is approximately 25% of the bus clock frequency.
50	SEL_TBCLK	Select time base clock. If the time base is enabled, this field functions as follows: 0 Time base is based on the processor clock 1 Time base is based on the TBCLK (RTC) input
51–62	—	Reserved, should be cleared.
63	NOPTI	No-op the data and instruction cache touch instructions. 0 **dcbt**, **dcbtst**, and **icbt** are enabled, as defined by the EIS. On the e500, if CT = 0, **icbt** is always a no-op, regardless of the value of NOPTI. If CT = 1, **icbt** does a touch load to an L2 cache, if one is present. 1 **dcbt**, **dcbtst**, and **icbt** are treated as no-ops; **dcblc** and **dcbtls** are not.

6.10.2 Hardware Implementation-Dependent Register 1 (HID1)

	32 33	34 39	40 45	46	47 49	50	51	52 63
Field	PLL_MODE	PLL_CFG	—	RFXE	—	ASTME	ABE	—
Reset	01[PLL_CFG]_0000_0000_0000_0000_0000_0000							
R/W	R/W							
SPR	SPR 1009							

Figure 6-32. Hardware Implementation-Dependent Register 1 (HID1)

Table 6-17. HID1 Field Descriptions

Bits	Name	Description
32–33	PLL_MODE	Read-only for integrated devices. 01 Fixed value for MPC8560
34–39	PLL_CFG	Reflected directly from configuration input pins (read-only). PLL_CFG[0–4] corresponds to the integer divide ratio and PLL_CFG5 is the half-mode bit. MPC8560 supports the following: 00010 0 ratio of 2:1 00010 1 ratio of 5:2 (2.5:1) 00011 0 ratio of 3:1 00011 1 ratio of 7:2 (3.5:1) Note that this value is also reflected to PORPLLSR[e500_Ratio]. See Section 18.4.1.1, “POR PLL Status Register (PORPLLSR).”
40–45	—	Reserved, should be cleared.

Table 6-17. HID1 Field Descriptions (continued)

Bits	Name	Description
46	RFXE	Read fault exception enable. Controls whether assertion of *core_fault_in* causes a machine check interrupt. The assertion of *core_fault_in* can result from an L2 multibit ECC error. It can also occur for a system error if logic on the integrated device signals a fault for nonfatal errors (read data is corrupt (or zero), but the bus transaction can complete without corrupting other system state, making it unnecessary to take a machine check (for example, a master abort of a PCI transaction). 0 Assertion of *core_fault_in* cannot cause a machine check. RFXE should be left clear if an interrupt is to be reported by the integrated device through *int* or *c_int* for this condition. If RFXE = 0, it is important that the integrated device generates an interrupt if *core_fault_in* is asserted. If *core_fault_in* is asserted, any data on the CCB is dropped, stalling the load/store unit and eventually causing the e500 pipeline either to stall until an interrupt occurs (typically generated by the programmable interrupt controller (PIC) in response to the fault) or to continue processing with bad data untill the interrupt occurs. Because *core_fault_in* cannot cause a machine check, if RFXE is 0, it is critical that the system be configured to generate the appropriate interrupt. It is also possible to hang the processor (requiring a hard reset to recover) if a guarded load hits in the L2 cache and gets an uncorrectable ECC error. Because of this, avoid defining memory as cacheable but guarded. If this combination is required, RFXE must be enabled, in which case an error causes both a machine check interrupt and an external interrupt when a bus fault condition is detected unless interrupts are masked for all sources of bus faults, such as DRAM ECC errors, PCI parity errors, local bus parity errors, and others. RFXE must also be set if software requires that code execution stop immediately when a bus fault occurs rather than continuing with the bad data until the interrupt arrives. Again, this results in both a machine check interrupt and an external interrupt when a bus fault is detected, unless all possible sources for bus fault have their interrupts masked.þThe machine check interrupt can then reenable normal interrupts and wait for the interrupt due to the fault to be received before returning from the machine check. 1 A machine check can occur due to assertion of *core_fault_in*. If MSR[ME] = 1 and a fault is signaled, a machine check interrupt occurs. If MSR[ME] = 0 and a fault is signaled, a checkstop occurs. Note that if RFXE is set and another mechanism is configured to generate an interrupt in response to assertion of *core_fault_in*, the same event causes two interrupts, the machine check enabled by setting RFXE and the interrupt triggered by the on-chip peripheral or other block; therefore, RFXE should be set only if no other mechanism is configured to generate an interrupt for this case. Note that the L2 cache detects any assertion of *core_fault_in* and ensures that the L2 cache is not corrupted when data is dropped for this type of transaction.
47–49	—	Reserved, should be cleared.
50	ASTME	Address bus streaming mode enable. This bit, along with the ECM stream control bits in the EEBACR, enables address bus streaming on the CCB. See Section 8.2.1.1, “ECM CCB Address Configuration Register (EEBACR).” 0 Address bus streaming mode disabled 1 Address bus streaming mode enabled
51	ABE	Address broadcast enable. The e500 broadcasts cache management instructions (**dcbst**, **dcblc** (CT = 1), **icblc** (CT = 1), **dcbf**, **dcbi**, **mbar**, **msync**, **tlbsync**, **icbi**) based on ABE. On the MPC8560**,** ABE must be set to allow management of external L2 caches. 0 Address broadcasting disabled 1 Address broadcasting enabled
52–63	—	Reserved, should be cleared.

6.11 L1 Cache Configuration Registers

6.11.1 L1 Cache Control and Status Register 0 (L1CSR0)

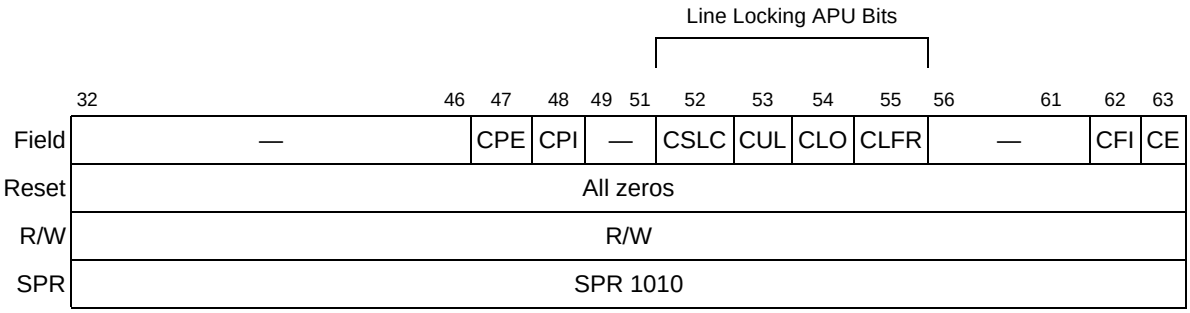


Figure 6-33. L1 Cache Control and Status Register 0 (L1CSR0)

Table 6-18. L1CSR0 Field Descriptions

Bits	Name	Description
32–46	—	Reserved, should be cleared.
47	CPE	(Data) Cache parity enable 0 Parity checking of the cache disabled 1 Parity checking of the cache enabled
48	CPI	(Data) Parity error injection enable 0 Parity error injection disabled 1 Parity error injection enabled. Cache parity must also be enabled (CPE = 1) when this bit is set.
49–51	—	Reserved, should be cleared.
52	CSLC	(Data) Cache snoop lock clear. Sticky bit set by hardware if a **dcbi** snoop (either internally or externally generated) invalidated a locked cache line. Note that the lock bit for that line is cleared whenever the line is invalidated. This bit can be cleared only by software. 0 The cache has not encountered an **dcbi** snoop that invalidated a locked line. 1 The cache has encountered an **dcbi** snoop that invalidated a locked line.
53	CUL	(Data) Cache unable to lock. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock set instruction was effective in the cache 1 Indicates a lock set instruction was not effective in the cache
54	CLO	(Data) Cache lock overflow. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock overflow condition was not encountered in the cache 1 Indicates a lock overflow condition was encountered in the cache
55	CLFR	(Data) Cache lock bits flash reset. Writing a 1 during a flash clear operation causes an undefined operation. Writing a 0 during a flash clear operation is ignored. Clearing occurs regardless of the enable (CE) value. 0 Default. 1 Hardware initiates a cache lock bits flash clear operation. This bit is cleared when the operation is complete.
56–61	—	Reserved, should be cleared.

Table 6-18. L1CSR0 Field Descriptions (continued)

Bits	Name	Description
62	CFI	(Data) Cache flash invalidate. 0 No cache invalidate. Writing a 0 to CFI during an invalidation operation is ignored. 1 Cache invalidation operation. A cache invalidation operation is initiated by hardware. Once complete, this bit is cleared. Writing a 1 during an invalidation operation causes an undefined operation. Invalidation occurs regardless of the enable (CE) value.
63	CE	(Data) Cache enable 0 The cache is neither accessed or updated. 1 Enables cache operation

6.11.2 L1 Cache Control and Status Register 1 (L1CSR1)

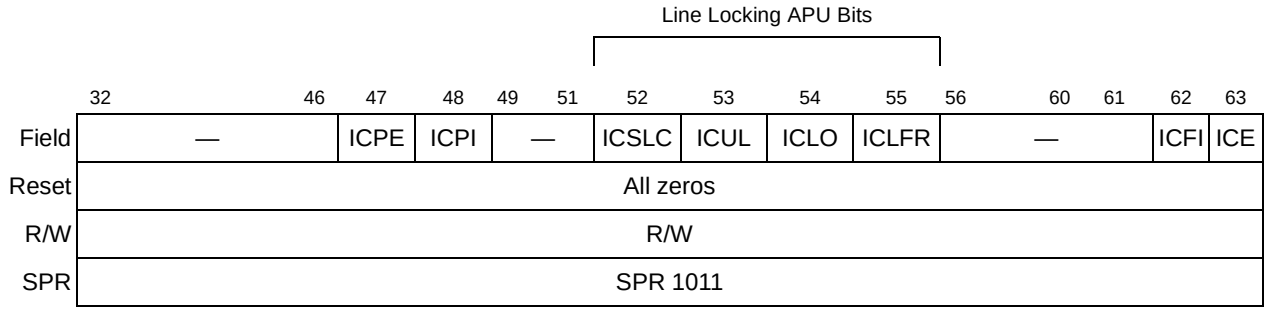


Figure 6-34. L1 Cache Control and Status Register 1 (L1CSR1)

Table 6-19. L1CSR1 Field Descriptions

Bits	Name	Description
32–46	—	Reserved, should be cleared.
47	ICPE	Instruction cache parity enable 0 Parity checking of the instruction cache disabled 1 Parity checking of the instruction cache enabled
48	ICPI	Instruction parity error injection enable 0 Parity error injection disabled 1 Parity error injection enabled. Note that instruction cache parity must also be enabled (ICPE = 1) when this bit is set.
49–51	—	Reserved, should be cleared.
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

Table 6-19. L1CSR1 Field Descriptions (continued)

Bits	Name	Description
54	ICLO	Instruction cache lock overflow. Sticky bit set by hardware and cleared by writing 0 to this bit location. 0 Indicates a lock overflow condition was not encountered in the instruction cache 1 Indicates a lock overflow condition was encountered in the instruction cache
55	ICLFR	Instruction cache lock bits flash reset. Writing 0 and then 1 flash clears the lock bit of all entries in the instruction cache; clearing occurs independently from the value of the enable bit (ICE). ICLFR is always read as 0.
56–61	—	Reserved, should be cleared.
62	ICFI	Instruction cache flash invalidate. Written to 0 and then 1 to flash clear the valid bit of all entries in the instruction cache; operates independently from the value of the enable bit (ICE). ICI is always read as 0.
63	ICE	Instruction cache enable 0 The instruction cache is neither accessed or updated. 1 Enables instruction cache operation

6.11.3 L1 Cache Configuration Register 0 (L1CFG0)

	32–33	34–38	39–40	41–42	43	44	45–49	50–52	53–55	56–63
Field	CARCH	—	CBSIZE	CREPL	CLA	CPA	—	CNWAY	—	CSIZE
Reset	00	00_000	00	01	1	1	000_00	11_1	000	0010_0000
R/W	Read only									
SPR	SPR 515									

Figure 6-35. L1 Cache Configuration Register 0 (L1CFG0)

Table 6-20. L1CFG0 Field Descriptions

Bits	Name	Description
32–33	CARCH	Cache architecture 00 Harvard 01 Unified
34–38	—	Reserved, should be cleared.
39–40	CBSIZE	Cache line size 0 32 bytes 1 64 bytes
41–42	CREPL	Cache replacement policy 0 True LRU 1 Pseudo LRU
43	CLA	Cache locking APU available 0 Unavailable 1 Available
44	CPA	Cache parity available 0 Unavailable 1 Available

Table 6-20. L1CFG0 Field Descriptions (continued)

Bits	Name	Description
45–49	—	Reserved, should be cleared.
50–52	CNWAY	Cache number of ways 111 Indicates 8 ways
53–55	—	Reserved, should be cleared.
56–63	CSIZE	Cache size 0x20 indicates 32 Kbytes

6.11.4 L1 Cache Configuration Register 1 (L1CFG1)

	32–38	39–40	41–42	43	44	45–52	53–63
Field	—	ICBSIZ	ICREPL	ICLA	ICPA	ICNWAY	ICSIZE
Reset	0000_000	0_0	01	1	1	000_0011_1	000_0010_0000
R/W	Read only						
SPR	SPR 516						

Figure 6-36. L1 Cache Configuration Register 1 (L1CFG1)

Table 6-21. L1CFG1 Field Descriptions

Bits	Name	Description
32–38	—	Reserved, should be cleared.
39–40	ICBSIZ	Instruction cache block size 00 Indicates block size of 32 bytes
41–42	ICREPL	Instruction cache replacement policy 01 Indicates pseudo-LRU policy
43	ICLA	Instruction cache locking available 1 Indicates available
44	ICPA	Instruction cache parity available 1 Indicates available
45–52	ICNWAY	Instruction cache number of ways 111 Indicates 8 ways
53–63	ICSIZE	Instruction cache size 0x20 indicates 32 Kbytes

6.12 MMU Registers

6.12.1 Process ID Registers (PID0–PID2)

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

Figure 6-37. Process ID Registers (PID0–PID2)

6.12.2 MMU Control and Status Register 0 (MMUCSR0)

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

Figure 6-38. MMU Control and Status Register 0 (MMUCSR0)

Table 6-22. MMUCSR0 Field Descriptions

Bits	Name	Description
32–60	—	Reserved, should be cleared.
61	L2TLB0_FI	TLB0 flash invalidate (write to 1 to invalidate)
62	L2TLB1_FI	TLB1 flash invalidate (write 1 to invalidate) 0 No flash invalidate. Writing a 0 to this bit during an invalidation operation is ignored. 1 TLB1 invalidation operation. Hardware initiates a TLB1 invalidation operation. When this operation is complete, this bit is cleared. Writing a 1 during an invalidation operation causes an undefined operation.
63	—	Reserved, should be cleared.

6.12.3 MMU Configuration Register (MMUCFG)

	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 6-39. MMU Configuration Register (MMUCFG)

Table 6-23. MMUCFG Field Descriptions

Bits	Name	Description
32–48	—	Reserved, should be cleared.
49–52	NPIDS	Number of PID registers, a 4-bit field that indicates the number of PID registers provided by the processor. The e500 implements three PIDs.
53–57	PIDSIZE	PID register size. The 5-bit value of PIDSIZE is one less than the number of bits in each of the PID registers implemented by the processor. The processor implements only the least significant PIDSIZE+1 bits in the PID registers. 00111 indicates 8-bit registers. This is the value presented by the e500.
58–59	—	Reserved, should be cleared.
60–61	NTLBS	Number of TLBs. The value of NTLBS is one less than the number of software-accessible TLB structures that are implemented by the processor. NTLBS is set to one less than the number of TLB structures so that its value matches the maximum value of MAS0[TLBSEL]. 00 1 TLB 01 2 TLBs. This is the value presented by the e500. 10 3 TLBs 11 4 TLBs
62–63	MAVN	MMU architecture version number. Indicates the version number of the architecture of the MMU implemented by the processor. 0b00 indicates version 1.0.

6.12.4 TLB Configuration Registers (TLB*n*CFG)

6.12.4.1 TLB0 Configuration Register 0 (TLB0CFG)

	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 6-40. TLB Configuration Register 0 (TLB0CFG)

Table 6-24. 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

Table 6-24. TLB0CFG Field Descriptions (continued)

Bits	Name	Description
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 TLB1 Configuration Register 1 (TLB1CFG)

	32–39	40–43	44–47	48	49	50–51	52–63
Field	ASSOC	MINSIZE	MAXSIZE	IPROT	AVAIL	—	NENTRY
e500 Reset Values	0001_0000	0001	1001	1	1	00	0000_0001_0000
R/W	Read only						
SPR	SPR 689						

Figure 6-41. TLB Configuration Register 1 (TLB1CFG)

Table 6-25. 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

6.12.5.1 MAS Register 0 (MAS0)

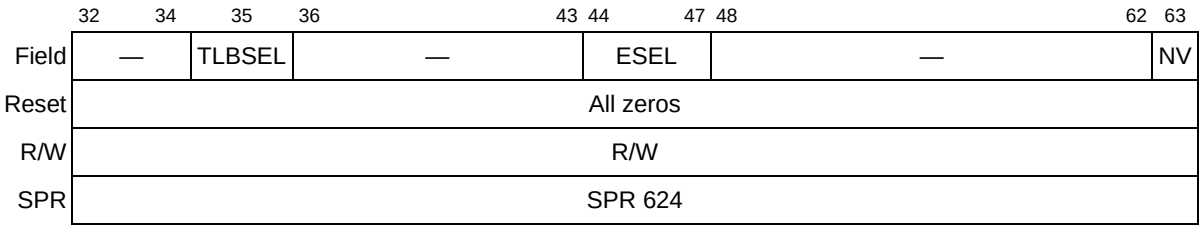


Figure 6-42. MAS Register 0 (MAS0)

Table 6-26. MAS0 Field Descriptions—MMU Read/Write and Replacement Control

Bits	Name	Descriptions
32–34	—	Reserved, should be cleared.
35	TLBSEL	Selects TLB for access 0 TLB0 1 TLB1
36–43	—	Reserved, should be cleared.
44–47	ESEL	Entry select. Number of entry in selected array to be used for **tlbwe**. This field is also updated on TLB error exceptions (misses), and **tlbsx** hit and miss cases. 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 from MAS2[EPN] 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.
48–62	—	Reserved, should be cleared.
63	NV	Next victim. Next victim bit value to be written to TLB0[NV] on execution of **tlbwe**. This field is also updated on TLB error exceptions (misses), **tlbsx** hit and miss cases and on execution of **tlbre**. This field is updated based on the calculated next victim bit for TLB0 (based on the round-robin replacement algorithm.) Note that this field is not defined for operations that specify TLB1 (when TLBSEL = 01).

6.12.5.2 MAS Register 1 (MAS1)

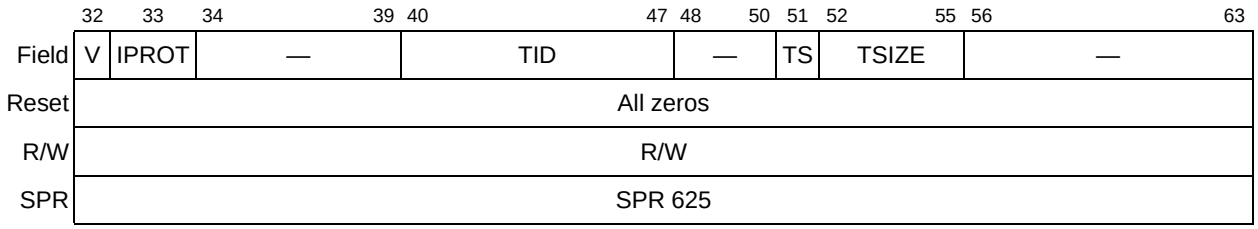


Figure 6-43. MAS Register 1 (MAS1)

Table 6-27. MAS1 Field Descriptions—Descriptor Context and Configuration Control

Bits	Name	Descriptions
32	V	TLB valid bit 0 This TLB entry is invalid. 1 This TLB entry is valid.
33	IPROT	Invalidate protect. Set to protect this TLB entry from invalidate operations due the execution of **tlbiva**[**x**] (TLB1 only). Note that not all TLB arrays are necessarily protected from invalidation with IPROT. Arrays that support invalidate protection are denoted as such in the TLB configuration registers. 0 Entry is not protected from invalidation 1 Entry is protected from invalidation.
34–39	—	Reserved, should be cleared.
40–47	TID	Translation identity. An 8-bit field that defines the process ID for this TLB entry. TID is compared with the current process IDs of the three virtual address to be translated. A TID value of 0 defines an entry as global and matches with all process IDs.
48–50	—	Reserved, should be cleared.
51	TS	Translation space. This bit is compared with the IS or DS fields of the MSR (depending on the type of access) to determine if this TLB entry may be used for translation.
52–55	TSIZE	Translation size. Defines the TLB entry page size. For arrays that contain fixed-size TLB entries, TSIZE is ignored. For variable page size arrays, the page size is 4^{TSIZE} Kbytes. Note that although the Freescale Semiconductor Book E standard supports all 16 page sizes defined in Book E, the e500 supports only the following: 0001 4 Kbyte 0010 16 Kbyte 0011 64 Kbyte 0100 256 Kbyte 0101 1 Mbyte 0110 4 Mbyte 0111 16 Mbyte 1000 64 Mbyte 1001 256 Mbyte
56–63	—	Reserved, should be cleared.

6.12.5.3 MAS Register 2 (MAS2)

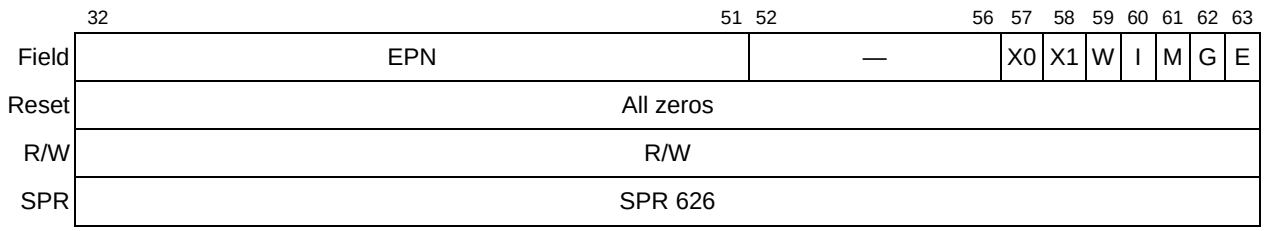


Figure 6-44. MAS Register 2 (MAS2)

Table 6-28. MAS2 Field Descriptions—EPN and Page Attributes

Bits	Name	Description
32–51	EPN	Effective page number. Depending on page size, only the bits associated with a page boundary are valid. Bits that represent offsets within a page are ignored and should be cleared.
52–56	—	Reserved for implementation-specific use
57	X0	Implementation-dependent page attribute
58	X1	Implementation-dependent page attribute
59	W	Write-through 0 This page is considered write-back with respect to the caches in the system. 1 All stores performed to this page are written through the caches to main memory.
60	I	Caching-inhibited 0 Accesses to this page are considered cacheable. 1 The page is considered caching-inhibited. All loads and stores to the page bypass the caches and are performed directly to main memory.
61	M	Memory 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 that miss in the L1 cache are performed without speculation (that is, they are known to be required). Speculative loads can be performed if they hit in the L1 cache. In addition, accesses to caching-inhibited pages are performed using only the memory element that is explicitly specified.
63	E	Endianness. Determines endianness for the corresponding page. Little-endian operation is true little endian, which differs from the modified little-endian byte-ordering model optionally available in previous devices that implement the original PowerPC architecture. See the *e500 Reference Manual* for more information on the Book E definition of endianness. 0 The page is accessed in big-endian byte order. 1 The page is accessed in true little-endian byte order.

6.12.5.4 MAS Register 3 (MAS3)

	32 … 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 6-45. MAS Register 3 (MAS3)

Table 6-29. MAS3 Field Descriptions–RPN and Access Control

Bits	Name	Description
32–51	RPN	Real page number. Depending on page size, only the bits associated with a page boundary are valid. Bits that represent offsets within a page are ignored and should be cleared.
52–53	—	Reserved, should be cleared.
54–57	U0–U3	User attribute bits. Associated with a TLB entry and can be used by system software. For example, they can hold information useful to a page-scanning algorithm or mark more abstract page attributes.
58–63	PERMIS	Permission bits (UX, SX, UW, SW, UR, SR). User and supervisor read, write, and execute permission bits.

6.12.5.5 MAS Register 4 (MAS4)

	32–34	35	36–45	46–47	48–51	52–55	56	57	58	59	60	61	62	63
Field	—	TLBSELD	—	TIDSELD	—	TSIZED	—	X0D	X1D	WD	ID	MD	GD	ED
Reset	All zeros													
R/W	R/W													
SPR	SPR 628													

Figure 6-46. MAS Register 4 (MAS4)

Table 6-30. MAS4 Field Descriptions—Hardware Replacement Assist Configuration

Bits	Name	Description
32–34	—	Reserved, should be cleared.
35	TLBSELD	TLBSEL default value. The default value to be loaded in MAS0[TLBSEL] on a TLB miss exception. 0 TLB0 1 TLB1
36–45	—	Reserved, should be cleared.
46–47	TIDSELD	TID default selection value. A 2-bit field that specifies which of the current PID registers should be used to load the MAS1[TID] field on a TLB miss exception.
		The e500 implementation defines this field as follows: 00 PID0 01 PID1 10 PID2 11 TIDZ (0x00) (all zeros)
48–51	—	Reserved, should be cleared.
52–55	TSIZED	Default TSIZE value. Specifies the default value to be loaded into MAS1[TSIZE] on a TLB miss exception.
56	—	Reserved, should be cleared.
57	X0D	Default X0 value. Specifies the default value to be loaded into MAS2[X0] on a TLB miss exception.
58	X1D	Default X1 value. Specifies the default value to be loaded into MAS2[X1] on a TLB miss exception.
59	WD	Default W value. Specifies the default value to be loaded into MAS2[W] on a TLB miss exception.

Table 6-30. MAS4 Field Descriptions—Hardware Replacement Assist Configuration (continued)

Bits	Name	Description
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)

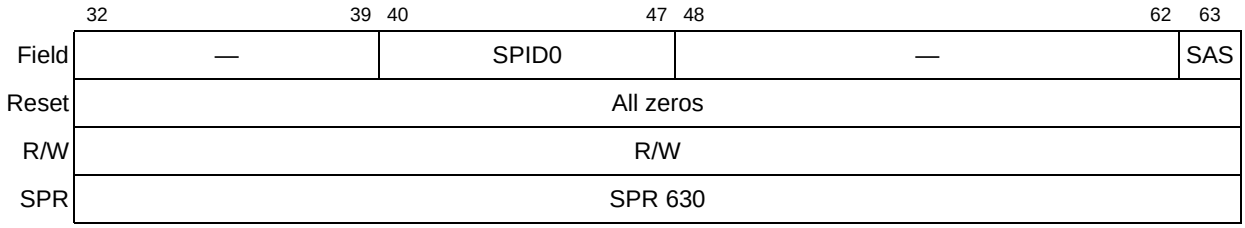


Figure 6-47. MAS Register 6 (MAS6)

Table 6-31. MAS6—TLB Search Context Register 0

Bits	Name	Comments, or Function when Set
32–39	—	Reserved, should be cleared.
40–47	SPID0	Specifies the PID value (recent value of PID0) used when searching the TLB during execution of **tlbsx**.
48–62	—	Reserved, should be cleared.
63	SAS	Address space (AS) value for searches. Specifies the value of AS used when searching the TLB (during execution of **tlbsx**).

6.13 Debug Registers

6.13.1 Debug Control Registers (DBCR0–DBCR2)

6.13.1.1 Debug Control Register 0 (DBCR0)

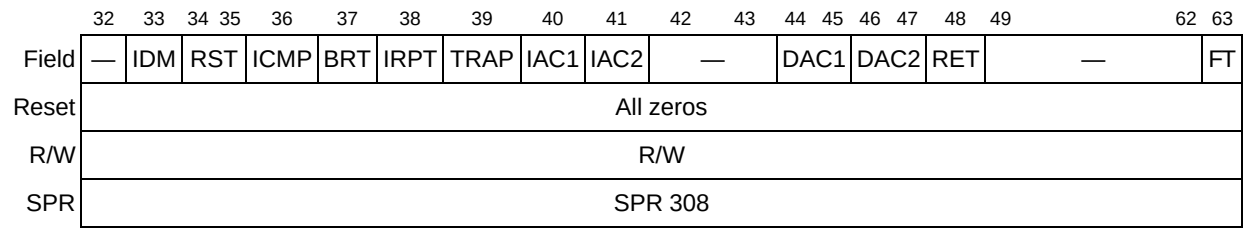


Figure 6-48. Debug Control Register 0 (DBCR0)

Table 6-32. DBCR0 Field Descriptions

Bits	Name	Description
32	—	Reserved, should be cleared.
33	IDM	Internal debug mode 0 Debug interrupts are disabled. No debug interrupts are taken and debug events are not logged. 1 If MSR[DE] = 1, the occurrence of a debug event or the recording of an earlier debug event in the DBSR when MSR[DE] = 0 or DBCR0[IDM] = 0 causes a debug interrupt. **Programming note:** Software must clear debug event status in the DBSR in the debug interrupt handler when a debug interrupt is taken before re-enabling interrupts through MSR[DE]. Otherwise, redundant debug interrupts are taken for the same debug event.
34–35	RST	Reset. 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
39	TRAP	Trap debug event enable 0 TRAP debug events cannot occur 1 TRAP debug events can occur
40	IAC1	Instruction address compare 1 debug event enable 0 IAC1 debug events cannot occur 1 IAC1 debug events can occur
41	IAC2	Instruction address compare 2 debug event enable 0 IAC2 debug events cannot occur 1 IAC2 debug events can occur
42–43	—	Reserved, should be cleared.
44–45	DAC1	Data address compare 1 debug event enable 00 DAC1 debug events cannot occur 01 DAC1 debug events can occur only if a store-type data storage access 10 DAC1 debug events can occur only if a load-type data storage access 11 DAC1 debug events can occur on any data storage access
46–47	DAC2	Data address compare 2 debug event enable 00 DAC2 debug events cannot occur 01 DAC2 debug events can occur only if a store-type data storage access 10 DAC2 debug events can occur only if a load-type data storage access 11 DAC2 debug events can occur on any data storage access

Table 6-32. DBCR0 Field Descriptions (continued)

Bits	Name	Description
48	RET	Return debug event enable 0 RET debug events cannot occur 1 RET debug events can occur Note: An **rfci** does not cause an RET debug event if MSR[DE] = 0 at the time that **rfci** executes.
49–62	—	Reserved, should be cleared.
63	FT	Freeze timers on debug event 0 Enable clocking of timers 1 Disable clocking of timers if any DBSR bit is set (except MRR)

6.13.1.2 Debug Control Register 1 (DBCR1)

	32 33	34 35	36 37	38 39	40 41	42 63
Field	IAC1US	IAC1ER	IAC2US	IAC2ER	IAC12M	—
Reset	All zeros					
R/W	R/W					
SPR	SPR 309					

Figure 6-49. Debug Control Register 1 (DBCR1)

Table 6-33. DBCR1 Field Descriptions

Bits	Name	Description
32–33	IAC1US	Instruction address compare 1 user/supervisor mode 00 IAC1 debug events can occur 01 Reserved 10 IAC1 debug events can occur only if MSR[PR] = 0 11 IAC1 debug events can occur only if MSR[PR] = 1
34–35	IAC1ER	Instruction address compare 1 effective/real mode 00 IAC1 debug events are based on effective addresses 01 Reserved on the e500 10 IAC1 debug events are based on effective addresses and can occur only if MSR[IS] = 0 11 IAC1 debug events are based on effective addresses and can occur only if MSR[IS] = 1
36–37	IAC2US	Instruction address compare 2 user/supervisor mode 00 IAC2 debug events can occur 01 Reserved 10 IAC2 debug events can occur only if MSR[PR] = 0 11 IAC2 debug events can occur only if MSR[PR] = 1
38–39	IAC2ER	Instruction address compare 2 effective/real mode 00 IAC2 debug events are based on effective addresses 01 Reserved on the e500 10 IAC2 debug events are based on effective addresses and can occur only if MSR[IS] = 0 11 IAC2 debug events are based on effective addresses and can occur only if MSR[IS] = 1

Table 6-33. DBCR1 Field Descriptions (continued)

Bits	Name	Description
40–41	IAC12M	Instruction address compare 1/2 mode 00 Exact address compare. IAC1 debug events can occur only if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can occur only if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 and IAC2 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1,plus ANDed with the contents of IAC2. If IAC1US≠IAC2US or IAC1ER≠IAC2ER, results are boundedly undefined. 10 Inclusive address range compare. IAC1 and IAC2 debug events 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.
42–63	—	Reserved, should be cleared.

6.13.1.3 Debug Control Register 2 (DBCR2)

	32–33	34–35	36–37	38–39	40–41	42–63
Field	DAC1US	DAC1ER	DAC2US	DAC2ER	DAC12M	—
Reset	All zeros					
R/W	R/W					
SPR	SPR 310					

Figure 6-50. Debug Control Register 2 (DBCR2)

Table 6-34. DBCR2 Field Descriptions

Bits	Name	Description
32–33	DAC1US	Data address compare 1 user/supervisor mode 00 DAC1 debug events can occur 01 Reserved 10 DAC1 debug events can occur only if MSR[PR] = 0. 11 DAC1 debug events can occur only if MSR[PR] = 1.
34–35	DAC1ER	Data address compare 1 effective/real mode 00 DAC1 debug events are based on effective addresses. 01 Reserved on the e500 10 DAC1 debug events are based on effective addresses and can occur only if MSR[DS] = 0. 11 DAC1 debug events are based on effective addresses and can occur only if MSR[DS] = 1.
36–37	DAC2US	Data address compare 2 user/supervisor mode 00 DAC2 debug events can occur. 01 Reserved 10 DAC2 debug events can occur only if MSR[PR] = 0. 11 DAC2 debug events can occur only if MSR[PR] = 1.

Table 6-34. DBCR2 Field Descriptions (continued)

Bits	Name	Description
38–39	DAC2ER	Data address compare 2 effective/real mode 00 DAC2 debug events are based on effective addresses. 01 Reserved on the e500 10 DAC2 debug events are based on effective addresses and can occur only if MSR[DS] = 0. 11 DAC2 debug events are based on effective addresses and can occur only if MSR[DS] = 1.
40–41	DAC12M	Data address compare 1/2 mode 00 Exact address compare. DAC1 debug events can occur only if the address of the data storage access is equal to the value specified in DAC1. DAC2 debug events can occur only if the address of the data storage access is equal to the value specified in DAC2. 01 Address bit match. DAC1 and DAC2 debug events can occur only if the address of the data storage access, ANDed with the contents of DAC2 are equal to the contents of DAC1, also ANDed with the contents of DAC2. If DAC1US ≠ DAC2US or DAC1ER ≠ DAC2ER, results are boundedly undefined. 10 Inclusive address range compare. DAC1 and DAC2 debug events can occur only if the address of the data storage access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. If DAC1US ≠ DAC2US or DAC1ER ≠ DAC2ER, results are boundedly undefined. 11 Exclusive address range compare. DAC1 and DAC2 debug events can occur only if the address of the data storage access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. If DAC1US ≠ DAC2US or DAC1ER ≠ DAC2ER, results are boundedly undefined.
42–63	—	Reserved, should be cleared.

6.13.2 Debug Status Register (DBSR)

	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	—	DAC1R	DAC1W	DAC2R	DAC2W	RET	—
Reset	00		Undef.	0000_0000_0000												
R/W	R/W															
SPR	SPR 304															

Figure 6-51. Debug Status Register (DBSR)

Table 6-35. DBSR Field Descriptions

Bits	Name	Description
32	IDE	Imprecise debug event. Set if MSR[DE] = 0 and a debug event causes its respective DBSR bit to be set. Functions as write-one-to-clear.
33	UDE	Unconditional debug event. Set if an unconditional debug event occurred. Functions as write-one-to-clear. If $\overline{\text{UDE}}$ (level sensitive, active low) is asserted, DBSR[UDE] is affected as follows: MSR[DE] DBCR0[IDM] Action X 0 No action. 0 1 UDE is set. 1 1 UDE is set and a debug interrupt is taken.

Table 6-35. DBSR Field Descriptions (continued)

Bits	Name	Description
34–35	MRR	Most recent reset. Functions as write-one-to-clear. 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.
36	ICMP	Instruction complete debug event. Set if an instruction completion debug event occurred and DBCR0[ICMP] = 1. Functions as write-one-to-clear.
37	BRT	Branch taken debug event. Set if a branch taken debug event occurred (DBCR0[BRT] = 1). Functions as write-one-to-clear.
38	IRPT	Interrupt taken debug event. Set if an interrupt taken debug event occurred (DBCR0[IRPT] = 1). Functions as write-one-to-clear.
39	TRAP	Trap instruction debug event. Set if a trap Instruction debug event occurred (DBCR0[TRAP] = 1). Functions as write-one-to-clear.
40	IAC1	Instruction address compare 1 debug event. Set if an IAC1 debug event occurred (DBCR0[IAC1] = 1). Functions as write-one-to-clear.
41	IAC2	Instruction address compare 2 debug event. Set if an IAC2 debug event occurred (DBCR0[IAC2] = 1). Functions as write-one-to-clear.
42–43	—	Reserved, should be cleared
44	DAC1R	Data address compare 1 read debug event. Set if a read-type DAC1 debug event occurred (DBCR0[DAC1] = 10 or 11). Functions as write-one-to-clear.
45	DAC1W	Data address compare 1 write debug event. Set if a write-type DAC1 debug event occurred (DBCR0[DAC1] = 01 or 11). Functions as write-one-to-clear.
46	DAC2R	Data address compare 2 read debug event.Set if a read-type DAC2 debug event occurred (DBCR0[DAC2] = 10 or 11). Functions as write-one-to-clear.
47	DAC2W	Data address compare 2 write debug event. Set if a write-type DAC2 debug event occurred (DBCR0[DAC2] = 01 or 11). Functions as write-one-to-clear.
48	RET	Return debug event. Set if a return debug event occurred (DBCR0[RET] = 1). Functions as write-one-to-clear.
49–63	—	Reserved, should be cleared.

6.13.3 Instruction Address Compare Registers (IAC1–IAC2)

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

Figure 6-52. Instruction Address Compare Registers (IAC1–IAC2)

6.13.4 Data Address Compare Registers (DAC1–DAC2)

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

Figure 6-53. Data Address Compare Registers (DAC1–DAC2)

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

	High-Word Error Bits									Status Bits					
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Field	SOVH	OVH	FGH	FXH	FINVH	FDBZH	FUNFH	FOVFH	—		FINXS	FINVS	FDBZS	FUNFS	FOVFS	MODE
Reset	0000_0000_0000_0000															
R/W	R/W															

	Enable Bits															
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Field	SOV	OV	FG	FX	FINV	FDBZ	FUNF	FOVF	—	FINXE	FINVE	FDBZE	FUNFE	FOVFE	FRMC	
Reset	0000_0000_0000_0000															
R/W	R/W															
SPR	SPR 512															

Figure 6-54. Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)

Table 6-36. 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.

Table 6-36. SPEFSCR Field Descriptions (continued)

Bits	Name	Function
36	FINVH	Embedded floating-point invalid operation error high. Set when an input value on the high side is a NaN, Inf, or Denorm. Also set on a divide if both the dividend and divisor are zero.
37	FDBZH	Embedded floating-point divide by zero error high. Set if the dividend is non-zero and the divisor is zero.
38	FUNFH	Embedded floating-point underflow error high
39	FOVFH	Embedded floating-point overflow error high
40–41	—	Reserved, should be cleared.
42	FINXS	Embedded floating-point inexact sticky. FINXS = FINXS | FGH | FXH | FG | FX
43	FINVS	Embedded floating-point invalid operation sticky. Location for software to use when implementing true IEEE floating point.
44	FDBZS	Embedded floating-point divide by zero sticky. FDBZS = FDBZS | FDBZH | FDBZ
45	FUNFS	Embedded floating-point underflow sticky. Storage location for software to use when implementing true IEEE floating point.
46	FOVFS	Embedded floating-point overflow sticky. Storage location for software to use when implementing true IEEE floating point.
47	MODE	Embedded floating-point mode (read only on e500)
48	SOV	Integer summary overflow. Set whenever an SPE instruction (except **mtspr**) sets OV. SOV remains set until it is cleared by **mtspr[SPEFSCR]**.
49	OV	Integer overflow. An overflow occurred in the lower half of the register while a SPE integer instruction is being executed.
50	FG	Embedded floating-point guard bit. Floating-point guard bit from the lower half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
51	FX	Embedded floating-point sticky bit. Floating bit from the lower half. The value is undefined if the processor takes a floating-point exception due to input error, floating-point overflow, or floating-point underflow.
52	FINV	Embedded floating-point invalid operation error. Set when an input value on the high side is a NaN, Inf, or Denorm. Also set on a divide if both the dividend and divisor are zero.
53	FDBZ	Embedded floating-point divide by zero error. Set of the dividend is non-zero and the divisor is zero.
54	FUNF	Embedded floating-point underflow error
55	FOVF	Embedded floating-point overflow error
56	—	Reserved, should be cleared.
57	FINXE	Embedded floating-point inexact enable
58	FINVE	Embedded floating-point invalid operation/input error exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FINV or FINVH is set by a floating-point instruction.

Table 6-36. SPEFSCR Field Descriptions (continued)

Bits	Name	Function
59	FDBZE	Embedded floating-point divide-by-zero exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FDBZ or FDBZH is set by a floating-point instruction.
60	FUNFE	Embedded floating-point underflow exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FUNF or FUNFH is set by a floating-point instruction.
61	FOVFE	Embedded floating-point overflow exception enable 0 Exception disabled 1 Exception enabled If the exception is enabled, a floating-point data exception is taken if FOVF or FOVFH is set by a floating-point instruction.
62–63	FRMC	Embedded floating-point rounding mode control 00 Round to nearest 01 Round toward zero 10 Round toward +infinity 11 Round toward –infinity

6.14.1 Accumulator (ACC)

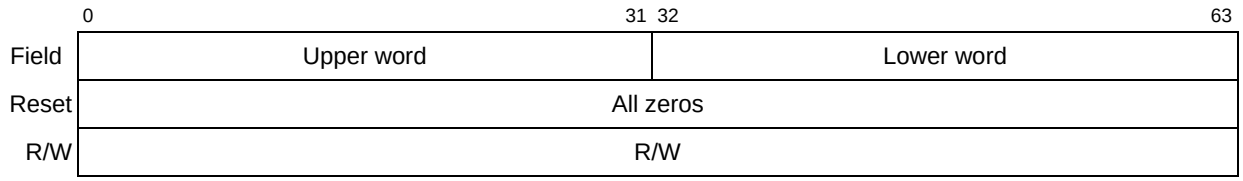


Figure 6-55. Accumulator (ACC)

Table 6-37. ACC FIeld Descriptions

Bits	Name	Function
0–31	Upper word	Holds the upper-word accumulate value for SPE multiply with accumulate instructions
32–63	Lower word	Holds the lower-word accumulate value for SPE multiply with accumulate instructions

6.15 Performance Monitor Registers (PMRs)

Table 6-38. Supervisor-Level PMRs (PMR[5] = 1)

Abbreviation	Register Name	PMR Number	pmr[0–4]	pmr[5–9]	Section/ Page
PMC0	Performance monitor counter 0	16	00000	10000	6.15.4/6-51
PMC1	Performance monitor counter 1	17	00000	10001	
PMC2	Performance monitor counter 2	18	00000	10010	
PMC3	Performance monitor counter 3	19	00000	10011	
PMGC0	Performance monitor global control register 0	400	01100	10000	6.15.1/6-49
PMLCa0	Performance monitor local control a0	144	00100	10000	6.15.2/6-49
PMLCa1	Performance monitor local control a1	145	00100	10001	
PMLCa2	Performance monitor local control a2	146	00100	10010	
PMLCa3	Performance monitor local control a3	147	00100	10011	
PMLCb0	Performance monitor local control b0	272	01000	10000	6.15.3/6-50
PMLCb1	Performance monitor local control b1	273	01000	10001	
PMLCb2	Performance monitor local control b2	274	01000	10010	
PMLCb3	Performance monitor local control b3	275	01000	10011	

Table 6-39. User-Level PMRs (PMR[5] = 0) (Read Only)

Abbreviation	Register Name	PMR Number	pmr[0–4]	pmr[5–9]	Section/ Page
UPMC0	User performance monitor counter 0	0	00000	00000	6.15.4/6-51
UPMC1	User performance monitor counter 1	1	00000	00001	
UPMC2	User performance monitor counter 2	2	00000	00010	
UPMC3	User performance monitor counter 3	3	00000	00011	
UPMLCa0	User performance monitor local control a0	128	00100	00000	6.15.3/6-50
UPMLCa1	User performance monitor local control a1	129	00100	00001	
UPMLCa2	User performance monitor local control a2	130	00100	00010	
UPMLCa3	User performance monitor local control a3	131	00100	00011	
UPMLCb0	User performance monitor local control b0	256	01000	00000	6.15.3/6-50
UPMLCb1	User performance monitor local control b1	257	01000	00001	
UPMLCb2	User performance monitor local control b2	258	01000	00010	
UPMLCb3	User performance monitor local control b3	259	01000	00011	
UPMGC0	User performance monitor global control register 0	384	01100	00000	6.15.2/6-49

6.15.1 Global Control Register 0 (PMGC0, UPMGC0)

	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 6-56. Performance Monitor Global Control Register 0 (PMGC0), User Performance Monitor Global Control Register 0 (UPMGC0)

Table 6-40. PMGC0 Field Descriptions

Bits	Name	Description
32	FAC	Freeze all counters. When FAC is set by hardware or software, PMLCx[FC] maintains its current value until it is changed by software. 0 The PMCs are incremented (if permitted by other PM control bits). 1 The PMCs are not incremented.
33	PMIE	Performance monitor interrupt enable 0 Performance monitor interrupts are disabled. 1 Performance monitor interrupts are enabled and occur when an enabled condition or event occurs.
34	FCECE	Freeze counters on enabled condition or event 0 The PMCs can be incremented (if permitted by other PM control bits). 1 The PMCs can be incremented (if permitted by other PM control bits) only until an enabled condition or event occurs. When an enabled condition or event occurs, PMGC0[FAC] is set. It is up to software to clear FAC.
35–63	—	Reserved, should be cleared.

6.15.2 Local Control A Registers (PMLCa0–PMLCa3, UPMLCa0–UPMLCa3)

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

Figure 6-57. Local Control A Registers (PMLCa0–PMLCa3), User Local Control A Registers (UPMLCa0–UPMLCa3)

Table 6-41. 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. These events are described in the *PowerPC e500 Core Reference Manual.*
48–63	—	Reserved, should be cleared.

6.15.3 Local Control B Registers (PMLCb0–PMLCb3, UPMLCb0–UPMLCb3)

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

Figure 6-58. Local Control B Registers (PMLCb0–PMLCb3), User Local Control B Registers (UPMLCb0–UPMLCb3)

Table 6-42. PMLCb0–PMLCb3 Field Descriptions

Bits	Name	Description
32–52	—	Reserved, should be cleared.
53–55	THRESHMUL	Threshold multiple 000 Threshold field is multiplied by 1 (PMLCb*n*[THRESHOLD] × 1) 001 Threshold field is multiplied by 2 (PMLCb*n*[THRESHOLD] × 2) 010 Threshold field is multiplied by 4 (PMLCb*n*[THRESHOLD] × 4) 011 Threshold field is multiplied by 8 (PMLCb*n*[THRESHOLD] × 8) 100 Threshold field is multiplied by 16 (PMLCb*n*[THRESHOLD] × 16) 101 Threshold field is multiplied by 32 (PMLCb*n*[THRESHOLD] × 32) 110 Threshold field is multiplied by 64 (PMLCb*n*[THRESHOLD] × 64) 111 Threshold field is multiplied by 128 (PMLCb*n*[THRESHOLD] × 128)
56–57	—	Reserved, should be cleared.
58–63	THRESHOLD	Threshold. Only events that exceed the threshold value are counted. Such events are implementation-dependent as are the dimension (for example duration in cycles) and granularity with which the value is interpreted. By varying the value, software can obtain a profile of the event characteristics subject to thresholding. For example, if PMC1 is configured to count cache misses that last longer than the threshold value, software can obtain the distribution of cache miss durations for a given program by monitoring the program repeatedly using a different threshold each time.

6.15.4 Performance Monitor Counter Registers (PMC0–PMC3, UPMC0–UPMC3)

	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 6-59. Performance Monitor Counter Registers (PMC0–PMC3), User Performance Monitor Counter Registers (UPMC0–UPMC3)

Table 6-43. PMC0–PMC3 Field Descriptions

Bits	Name	Description
32	OV	Overflow. When this bit is set, it indicates this counter reaches its maximum value.
33–63	Counter value	Indicates the number of occurrences of the specified event

Chapter 7
L2 Look-Aside Cache/SRAM

This chapter describes the organization of the on-chip L2/SRAM, cache coherency rules, cache line replacement algorithm, cache control instructions, and various cache operations. It also describes the interaction between the L2/SRAM and the e500 core complex.

7.1 L2 Cache Overview

The integrated 256-Kbyte L2 cache is organized as 1024 eight-way sets of 32-byte cache lines based on 32-bit physical addresses, as shown in Figure 7-1.

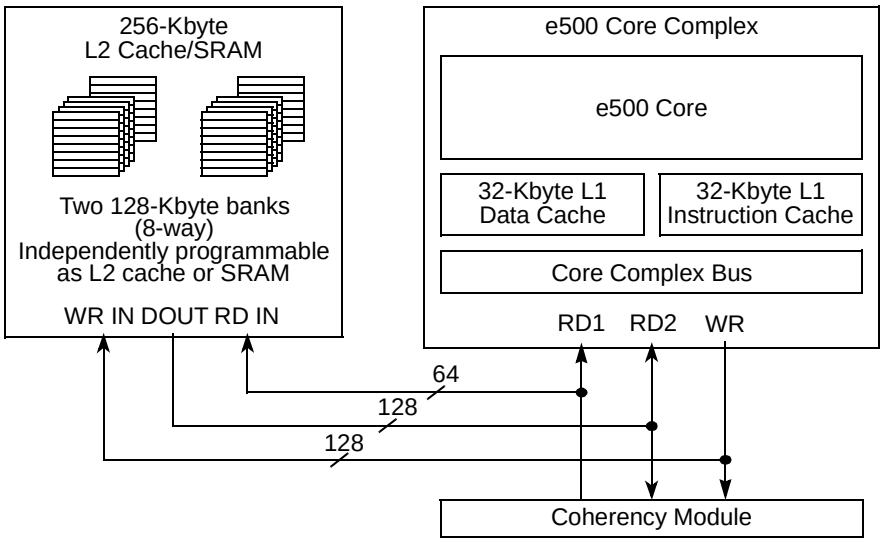


Figure 7-1. L2 Cache/SRAM Configuration

The SRAM can be configured with memory-mapped registers as externally accessible memory-mapped SRAM in addition to or instead of cache. The L2 cache can operate in the following modes, described in Section 7.2, “Cache Organization”:

- Full cache mode (256-Kbyte cache)
- Full memory-mapped SRAM mode (256-Kbyte SRAM mapped as a single 256-Kbyte block or two 128-Kbyte blocks)
- Half SRAM and half cache mode (128-Kbyte cache and 128-Kbyte memory-mapped SRAM)

7.1.1 L2 Cache and SRAM Features

Two 128-Kbyte arrays can be designated independently as externally-accessible memory-mapped SRAM or cache. The L2 cache has the following characteristics:

- Write-through, front-side cache
  - Front-side design provides easier cache access for I/O masters such as Ethernet, CPM, and RapidIO
  - Write-through design is more efficient on the processor bus for front-side caches
- Valid, locked, and stale states (no modified state)
- Two input data buses (64 and 128 bits) and one output data bus (128 bits wide)
- All accesses are fully pipelined and non-blocking (allows hits under misses)
- 256-Kbyte array organized as 1024 eight-way sets of 32-byte cache lines
- Eight-way set associativity (high level of associativity yields good performance even with many locked lines)
- Tag arrays contain 17 tag bits and 1 tag parity bit per line to support 256-Kbyte cache (1024 sets), or 18 tag bits to support 128-Kbyte cache (512 sets).
- Configurable to allocate processor instructions, data, or both
  - Allows external writes (stashing) to allocate and optionally lock a line using one of the two following methods:
    - Attributes attached to the transactions by initiator or ATMU
    - I/O devices can force memory writes to be allocated using programmed memory ranges
- Pseudo-LRU (7-bit replacement algorithm)
- Data ECC on 64-bit boundaries (single-error correction, double-error detection)
- Tag parity for 256-Kbyte mode (1 tag bit per line covering cache tags)
- Cache locking methods
  - Individual line locks are set and cleared by using e500 cache locking APU instructions—Data Cache Block Touch and Lock Set (**dcbtls**), Data Cache Block Touch for Store and Lock Set (**dcbtstls**), and Instruction Cache Block Touch and Lock Set (**icbtls**).
  - A lock attribute can be attached to write operations.
  - Individual line locks are set and cleared through core-initiated instructions, by external reads or writes, or by accesses to programmed memory ranges defined in L2 cache external write address registers (L2CEWAR*n*).
  - The entire cache can be locked by setting a configuration register appropriately.

- Lock clearing methods
  - Individual locks cleared by cache locking APU instructions (Instruction Cache Block Lock Clear (**icblc**) and Data Cache Block Lock Clear (**dcblc**)) or by snooped flush unless entire cache is locked
  - Flash clearing of all instruction and/or data locks is done by writes to configuration registers
  - An unlock attribute attached to a read instruction.
- Error injection modes for testing

SRAM features include the following:

- I/O devices access SRAM regions by marking transactions as snoopable (global)
- Regions can reside at any aligned location in the memory map
- For accesses of less than a cache-line, byte-accessible ECC is protected using read-modify-write transactions.

7.2 Cache Organization

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

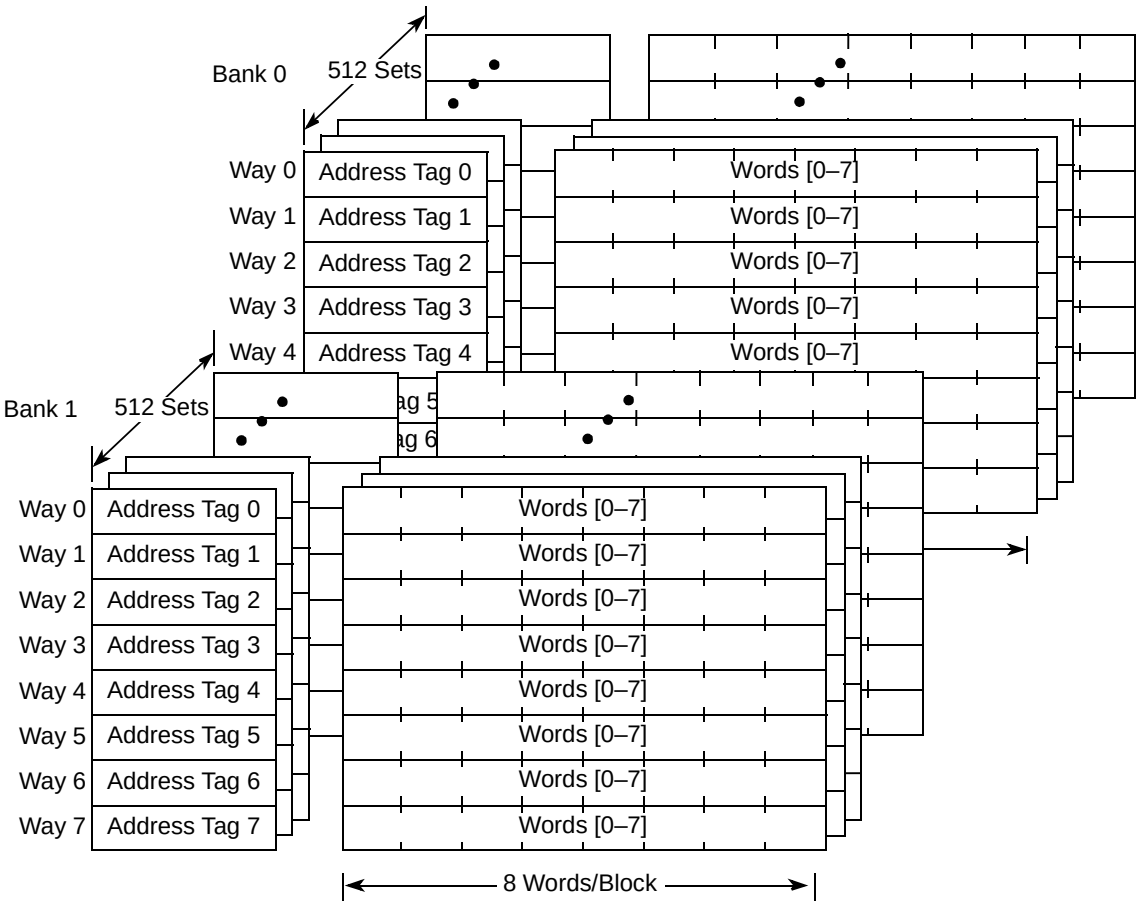


Figure 7-2. Cache Organization

The tag size depends on whether both 128-Kbyte arrays are configured as cache. Figure 7-3 shows how physical address bits are used to access the L2 in full cache mode (256-Kbyte cache).

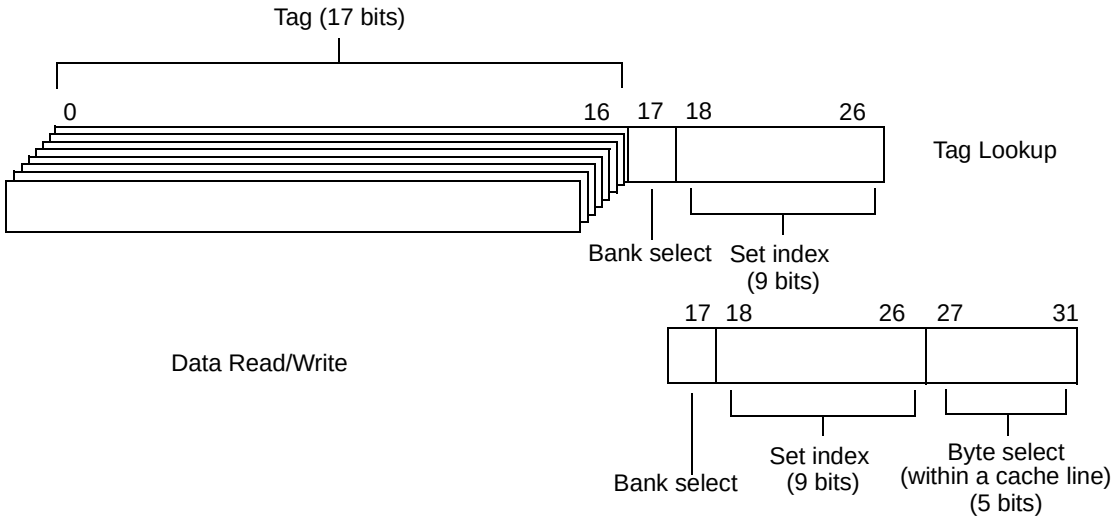


Figure 7-3. 256-Kbyte L2 Cache Address Configuration—Full Cache Mode

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. A match of a valid tag selects a 32-byte block of data within the set. Physical address bits 27–31 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.

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 (with no parity bit) 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.

As shown in Figure 7-4, the tag is 18 bits (with no parity bit) for cache accesses. For SRAM accesses, physical address bits 15–17 are used as the way select.

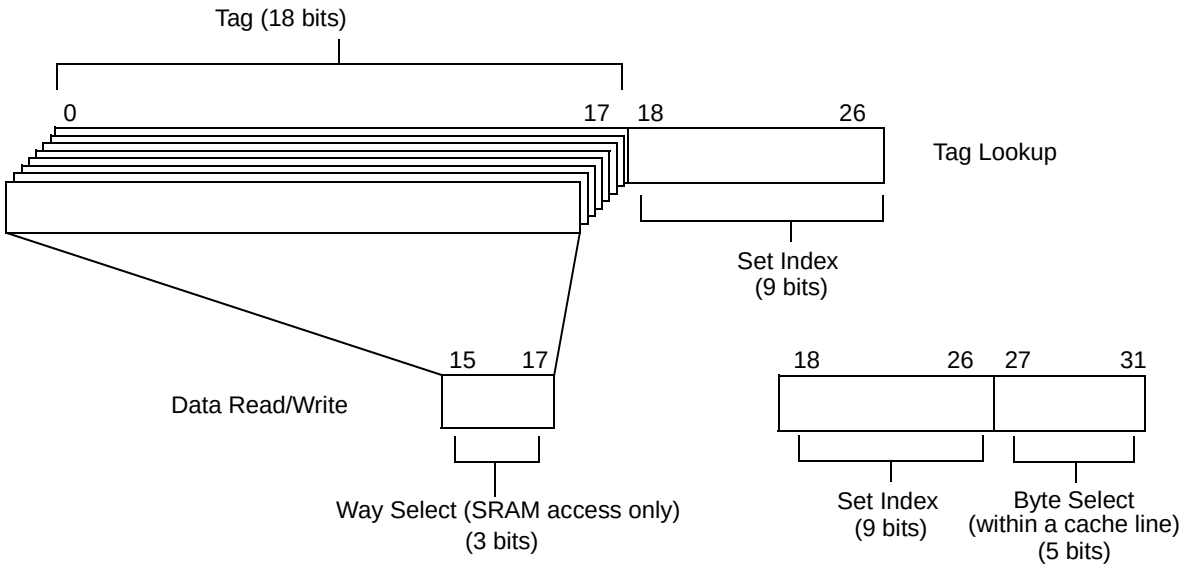


Figure 7-4. 128-Kbyte L2 Cache Address Configuration—Half SRAM, Half Cache Mode

The e500 core connects to the L2 cache and the system interface through the high-speed core complex bus (CCB). The e500 core and the L2 cache connect to the rest of the integrated device through the e500 coherency module (ECM). Figure 7-5 shows the data connections of the e500 core and L2/SRAM. The e500 core can simultaneously read 128 bits of data from the L2/SRAM, read 64 bits of data from the system interface, and write 128 bits of data to the L2/SRAM and/or system interface.

The L2/SRAM can be accessed by the e500 core or the system interface through the ECM. The L2 cache does not initiate transactions. Figure 7-5 shows the data bus connections of the e500 core and L2/SRAM.

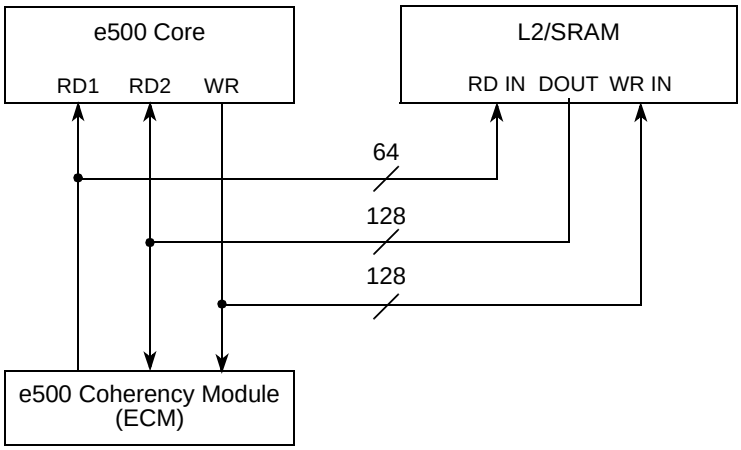


Figure 7-5. Data Bus Connection of CCB

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

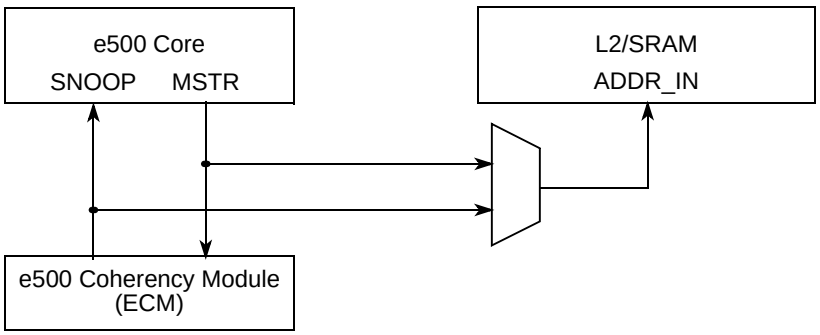


Figure 7-6. Address Bus Connection of CCB

In SRAM mode, if a non–cache-line read or write transaction is not preceded by a cache-line write, an ECC error occurs; such a non–cache-line write transaction cannot be allocated in the L2.

7.3 Memory Map/Register Definition

Table 7-1 shows the memory map for the L2/SRAM registers. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

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

Offset	Register	Access	Reset	Section/Page
0x2_0000	L2CTL—L2 control register	R/W	0x2000_0000	7.3.1.1/7-7
0x2_0010	L2CEWAR0—L2 cache external write address register 0	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0018	L2CEWCR0—L2 cache external write control register 0	R/W	0x0000_0000	7.3.1.3/7-11
0x2_0020	L2CEWAR1—L2 cache external write address register 1	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0028	L2CEWCR1—L2 cache external write control register 1	R/W	0x0000_0000	7.3.1.3/7-11

Table 7-1. L2/SRAM Memory-Mapped Registers (continued)

Offset	Register	Access	Reset	Section/Page
0x2_0030	L2CEWAR2—L2 cache external write address register 2	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0038	L2CEWCR2—L2 cache external write control register 2	R/W	0x0000_0000	7.3.1.3/7-11
0x2_0040	L2CEWAR3—L2 cache external write address register 3	R/W	0x0000_0000	7.3.1.2/7-10
0x2_0048	L2CEWCR3—L2 cache external write control register 3	R/W	0x0000_0000	7.3.1.3/7-11
0x2_0100	L2SRBAR0—L2 memory-mapped SRAM base address register 0	R/W	0x0000_0000	7.3.1.4/7-12
0x2_0108	L2SRBAR1—L2 memory-mapped SRAM base address register 1	R/W	0x0000_0000	7.3.1.4/7-12
0x2_0E00	L2ERRINJHI—L2 error injection mask high register	R/W	0x0000_0000	7.3.1.5.1/7-13
0x2_0E04	L2ERRINJLO—L2 error injection mask low register	R/W	0x0000_0000	7.3.1.5.1/7-13
0x2_0E08	L2ERRINJCTL—L2 error injection tag/ECC control register	R/W	0x0000_0000	7.3.1.5.1/7-13
0x2_0E20	L2CAPTDATAHI—L2 error data high capture register	R	0x0000_0000	7.3.1.5.2/7-15
0x2_0E24	L2CAPTDATALO—L2 error data low capture register	R	0x0000_0000	7.3.1.5.2/7-15
0x2_0E28	L2CAPTECC—L2 error syndrome register	R	0x0000_0000	7.3.1.5.2/7-15
0x2_0E40	L2ERRDET—L2 error detect register	Special	0x0000_0000	7.3.1.5.2/7-15
0x2_0E44	L2ERRDIS—L2 error disable register	R/W	0x0000_0000	7.3.1.5.2/7-15
0x2_0E48	L2ERRINTEN—L2 error interrupt enable register	R/W	0x0000_0000	7.3.1.5.2/7-15
0x2_0E4C	L2ERRATTR—L2 error attributes capture register	R/W	0x0000_0000	7.3.1.5.2/7-15
0x2_0E50	L2ERRADDR—L2 error address capture register	R	0x0000_0000	7.3.1.5.2/7-15
0x2_0E58	L2ERRCTL—L2 error control register	R/W	0x0000_0000	7.3.1.5.2/7-15

7.3.1 L2/SRAM Register Descriptions

The following sections describe registers that control and configure the L2/SRAM array.

7.3.1.1 L2 Control Register (L2CTL)

The L2 control register (L2CTL), shown in Figure 7-7, controls configuration and operation of the L2/SRAM array. The sequence for modifying L2CTL is as follows:

1. **mbar**
2. **isync**
3. **stw** (WIMG = 01xx) CCSRBAR+0x2_0000
4. **lwz** (WIMG = 01xx) CCSRBAR+0x2_0000
5. **mbar**

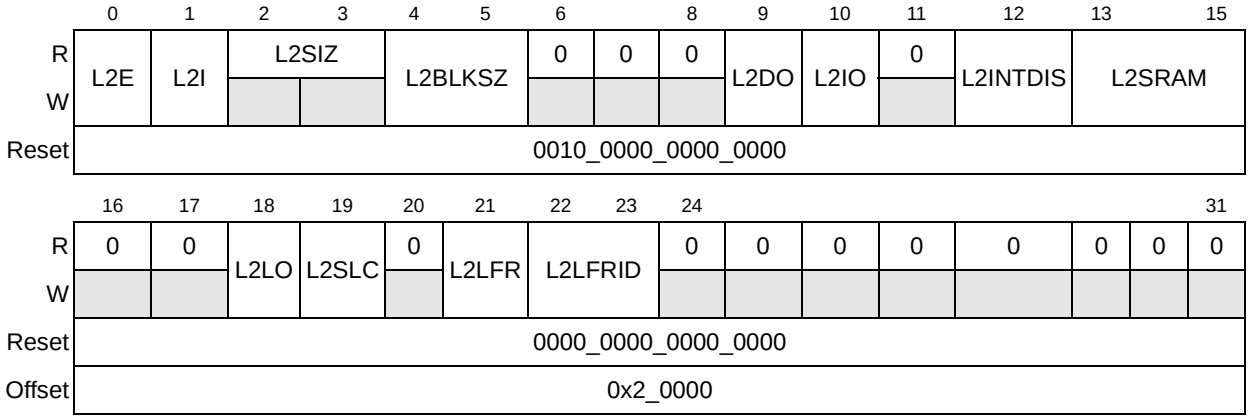


Figure 7-7. L2 Control Register (L2CTL)

Table 7-2 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. Note that L2I can be set regardless of the value of L2E.
1	L2I	L2 flash invalidate 0 The L2 status and LRU bits are not being cleared. 1 Setting L2I invalidates the L2 cache globally by clearing the all the L2 status bits, as well as the LRU algorithm. Memory-mapped SRAM is unaffected. Data in memory-mapped SRAM regions is 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. To change these bits, the L2 must be disabled (L2CTL[L2E] = 0). L2BLKSZ must be ≤ L2SIZ when L2E = 1. See the description of L2CTL[L2SRAM] for information on configuring the total SRAM as cache or memory-mapped SRAM. 00 Reserved 01 128 Kbyte 10 256 Kbyte 11 Reserved
6–8	—	Reserved

Table 7-2. L2CTL Field Descriptions (continued)

Bits	Name	Description
9	L2DO	L2 data-only. Reserved in full memory-mapped SRAM mode. L2DO may be changed while the L2 is enabled or disabled. 0 The L2 cache allocates entries for instruction fetches that miss in the L2. 1 The L2 cache allocates entries for processor data loads that miss in the L2 and for processor L1 castouts but does not allocate entries for instruction fetches that miss in the L2. Instruction accesses that hit in the L2, data accesses, and accesses from the system (including I/O stash writes) are unaffected. Note that if L2DO and L2IO are both set, no new lines are allocated into the L2 cache for any processor transactions, and processor writes and castouts that hit existing data in the cache invalidate those lines rather than updating them.
10	L2IO	L2 instruction-only. Reserved in full memory-mapped SRAM mode. Causes the L2 cache to allocate lines for instruction cache transactions only. L2IO may be changed while the L2 is enabled or disabled. 0 The L2 cache entries are allocated for data loads that miss in the L2 and for processor L1 castouts. 1 The L2 cache allocates entries for instruction fetch misses, but does not allocate entries for processor data transactions. Data accesses that hit in the L2, instruction accesses, and accesses from the system (including I/O stash writes) are unaffected. Note that if L2DO and L2IO are both set, no new lines are allocated into the L2 cache for any processor transactions, and processor writes and castouts that hit existing data in the cache invalidate those lines rather than updating them.
11	—	Reserved
12	L2INTDIS	Cache read intervention disable. Reserved for full memory-mapped SRAM mode. Used to disable cache read intervention. May be changed while the L2 is enabled or disabled. 0 Cache intervention is enabled. The ECM ensures that if a data read from another device hits in the L2 cache, it is serviced from the L2 cache. 1 Cache intervention is disabled
13–15	L2SRAM	L2 block assignment. Determines the L2 cache/memory-mapped SRAM block assignment. L2SIZ = L2BLKSIZ (1 block): 000 Block 0 = Cache 001 Block 0 = SRAM0 010–111 Reserved L2SIZ = L2BLKSIZ × 2 (2 blocks): Block 0 Block 1 000 Unused Cache 001 SRAM0 Unused 010 SRAM0 Cache 011 SRAM0 SRAM1 1xx Reserved To change these bits, the L2 must be disabled (L2CTL[L2E] = 0).
16–17	—	Reserved
18	L2LO	L2 cache lock overflow. Reserved in full memory-mapped SRAM mode. This sticky bit is set if an overlock condition is detected in the L2 cache. A lock overflow is triggered either by executing instruction or data cache block touch and lock set instructions or by performing L2 cache external writes with lock set. If all ways are locked and an attempt to stash is made, the stash is not allocated. 0 The L2 cache did not encounter a lock overflow. L2LO is cleared only by software. 1 The L2 cache encountered a lock overflow condition.

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 data cache line. Note that the lock bit for that line is cleared whenever the line is invalidated. L2SLC is reserved in full memory-mapped SRAM mode. 0 A snoop did not invalidate a locked L2 cache line. L2SLC is cleared only by software. 1 The L2 cache encountered a snoop that invalidated a locked line.
20	—	Reserved
21	L2LFR	L2 cache lock bits flash reset. The L2 cache must be enabled (L2CTL[L2E] = 1) for reset to occur. This field is reserved in full memory-mapped SRAM mode. 0 The L2 cache lock bits are not cleared or the clear operation completed. 1 A reset operation is issued that clears each L2 cache line's lock bits. Depending on the L2LFRID value, data or instruction locks, or both can be reset. Cache access is blocked during this time. After L2LFR is set, the L2 cache unit automatically clears L2LFR when the reset operation is complete (if L2CTL[L2E] is set).
22–23	L2LFRID	L2 cache lock bits flash reset select instruction or data. Indicates whether data or instruction lock bits or both are reset. 00 Not used 01 Reset data locks if L2LFR = 1 10 Reset instruction locks if L2LFR = 1 11 Reset both data and instruction locks if L2LFR = 1
24–31	—	Reserved

7.3.1.2 L2 Cache External Write Address Registers 0–3 (L2CEWAR*n*)

The MPC8560 supports allocating and locking of L2 cache lines from external agents, such as PCI. This functionality is called stashing. The L2 cache external write address registers 0–3 (L2CEWAR*n*) are paired with the L2 cache external write control registers 0–3 (L2CEWCR*n*) to control the cache external write functionality. Each register pair (for example, L2CEWAR0 and L2CEWCR0) specifies a programmed memory range that can be locked with a snoop write transaction. The address register must be naturally aligned to the window size in the corresponding control register. L2CEWAR*n* registers contain identical fields, as shown in Figure 7-8.

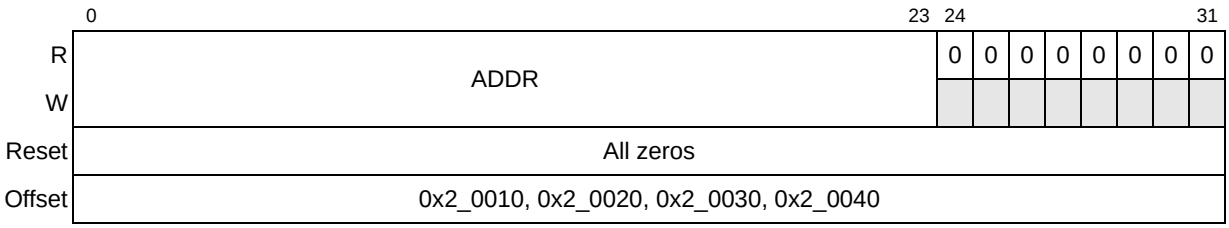


Figure 7-8. L2 Cache External Write Address Registers (L2CEWAR*n*)

Table 7-3 describes L2CEWAR*n* 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 0–3 (L2CEWCR*n*)

The L2CEWAR*n* registers are paired with the L2 cache external write control registers 0–3 (L2CEWCR*n*), shown in Figure 7-9, to control cache external write functionality.

	0	1	2					7	8	31
R	E	LOCK	0	0	0	0	0	0	SIZMASK	
W										
Reset	All zeros									
Offset	0x2_0018, 0x2_0028, 0x2_0038, 0x2_0048									

Figure 7-9. L2 Cache External Write Control Registers (L2CEWCR0–L2CEWCR3)

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

Table 7-4. L2CEWCR*n* Field Descriptions

Bits	Name	Description
0	E	External write enable. An external write matching the address window defined by L2CEWAR*n*/L2CEWCR*n* is allocated or updated in the L2 cache. 0 External writes for the L2CEWAR*n*/L2CEWCR*n* pair are disabled. 1 External writes are enabled for the L2CEWAR*n*/L2CEWCR*n* pair.
1	LOCK	Lock lines in the targeted cache. An external write matching the address window defined by L2CEWAR*n*/L2CEWCR*n* is locked in the L2 cache when it is allocated or updated. 0 The locked bit is not set when a line is allocated unless explicitly specified by transaction attributes. 1 Cache lines are allocated as locked. A hit to a valid, unlocked lines sets the lock.
2–7	—	Reserved

Table 7-4. L2CEWCR*n* Field Descriptions (continued)

Bits	Name	Description
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. Any value not listed below is illegal and produces boundedly undefined results. 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 Mbyte 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 Gbyte 1000 0000 0000 0000 0000 0000 2 Gbytes 0000 0000 0000 0000 0000 0000 4 Gbytes

7.3.1.4 L2 Memory-Mapped SRAM Base Address Registers 0–1 (L2SRBAR*n*)

The L2 memory-mapped SRAM base address registers (L2SRBAR*n*), shown in Figure 7-10, 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.

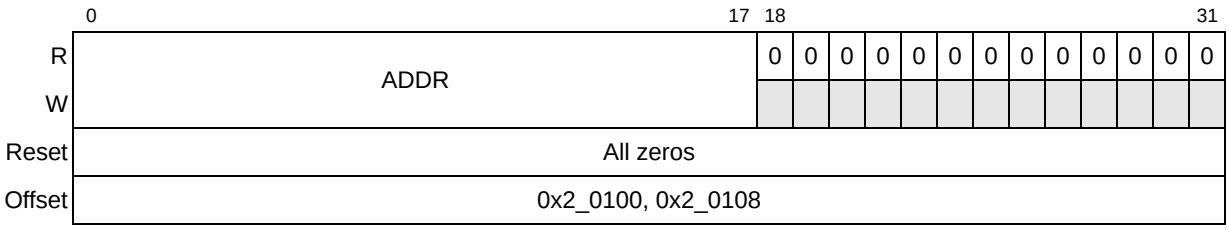


Figure 7-10. L2 Memory-Mapped SRAM Base Address Registers (L2SRBAR*n*)

L2SRBAR bits are described in Table 7-5.

Table 7-5. L2SRBAR*n* Field Descriptions

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

When enabled, the windows defined in L2SRBAR*n* supersede all other mappings of these addresses for processor and global (snoopable) I/O transactions. Therefore, SRAM windows must

never overlap configuration space as defined by CCSRBAR (see Section 3.3.1.1.2, “Configuration, Control, and Status Base Address Register (CCSRBAR)”). Overlapping SRAM and local access windows is discouraged because processor and snoopable I/O transactions would map to the SRAM while non-snooped I/O transactions would be mapped by the local access windows. Only if all accesses to the SRAM address range are snoopable can results be consistent if SRAM and local access windows overlap. Furthermore, L2SRBARn should not overlap a DDR DRAM space for which a valid chip select is defined. This could result in spurious ECC errors if ECC and speculative reads are enabled. See Section 2.2.3.7, “Illegal Interaction Between Local Access Windows and DDR SDRAM Chip Selects.”

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 information is saved only for the first one. Error reporting (by generating an interrupt) is enabled by setting the corresponding bit in the error interrupt enable register (L2ERRINTEN). Note that the error detect bit is set regardless of the state of the interrupt enable bit. When an error is detected, if error detection is enabled the L2 cache/SRAM always asserts an internal error signal with read data to prevent the L1 caches and architectural registers from being loaded with corrupt data. If error detection is disabled, the detected error bit is not set and no internal signal is asserted.

The L2 error detect register (L2ERRDET) is implemented as a bit-reset type register. Reading from this register occurs normally; however, write operations can clear but not set bits. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear bit 6 and not affect any other bits in the register, the value 0x0200_0000 is written to the register.

Note that in SRAM mode, if a non–cache-line read or write transaction is not preceded by a cache-line write, an ECC error occurs; such a non–cache-line write transaction cannot be allocated in the L2.

7.3.1.5.1 Error Injection Registers

The L2 cache includes support for injecting errors into the L2 data, data ECC, or tag. This may be used to test error recovery software by deterministically creating error scenarios.

The preferred method for error injection is to set all data pages to cache-inhibited (MMU TLB entry I = 1) except a scratch page, set L2CTL[L2DO] to prevent allocation of instruction accesses,

and invalidate the L2 by setting L2CTL[L2I] = 1. The following code sequence triggers an error, then detects it (A is an address in the scratch page):

```
dcbz A          | allocates the line in the L1 in the modified state
dcbtls_L2 A     | forces the line from the L1 and allocates the line in the L2
lwz A
```

Data or tag errors are injected into the line, according to the error injection settings in L2ERRINJHI, L2ERRINJLO, and L2ERRINJCTL, at allocation. The final load detects and reports the error (if enabled) and allows software to examine the offending data, address, and attributes.

Note that error injection enable bits in L2ERRINJCTL must be cleared by software and the L2 must be invalidated (by setting L2CTL[L2I]) before resuming L2 normal operation. Figure 7-11 shows the L2 error injection mask high register (L2ERRINJHI).

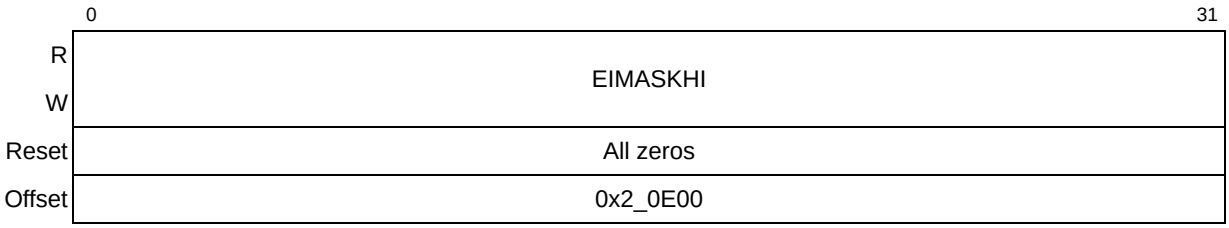


Figure 7-11. L2 Error Injection Mask High Register (L2ERRINJHI)

Table 7-6 describes L2ERRINJHI[EIMASKHI].

Table 7-6. L2ERRINJHI Field Description

Bits	Name	Description
0–31	EIMASKHI	Error injection mask/high word. A set bit corresponding to a data path bit causes that bit on the data path to be inverted on cache/SRAM writes if L2ERRINJCTL[DERRIEN] = 1.

Figure 7-12 shows the L2 error injection mask low register (L2ERRINJLO) fields.

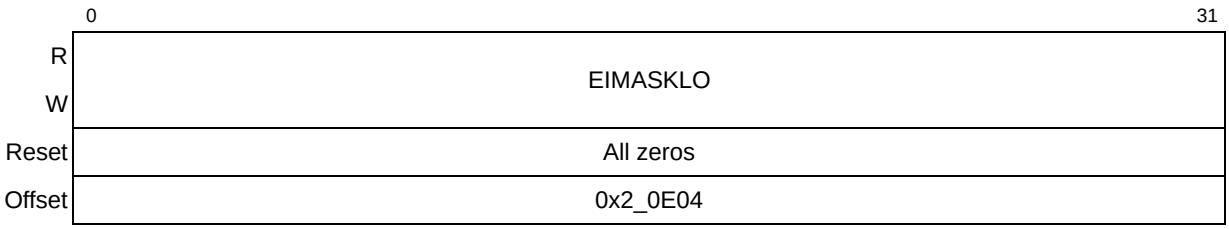


Figure 7-12. L2 Error Injection Mask Low Register (L2ERRINJLO)

Table 7-7 describes L2ERRINJLO[EIMASKLO].

Table 7-7. L2ERRINJLO Field Description

Bits	Name	Description
0–31	EIMASKLO	Error injection mask/low word. A set bit corresponding to a data path bit causes that bit on the data path to be inverted on SRAM writes if L2ERRINJCTL[DERRIEN] = 1.

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

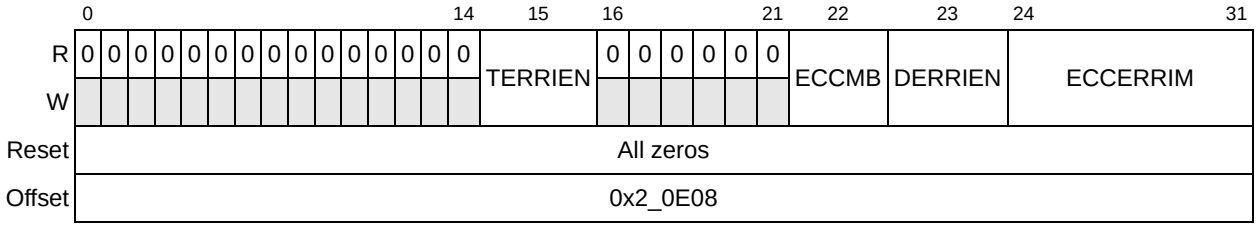


Figure 7-13. L2 Error Injection Mask Control Register (L2ERRINJCTL)

Table 7-8 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 Subsequent entries written to the L2 data array have data or ECC bits inverted as specified in the data and ECC error injection masks and/or data path byte mirrored onto ECC as specified by ECC mirror byte enable. Note: if both ECC mirror byte and data error injection are enabled, ECC mask error injection is performed on the mirrored ECC.
24–31	ECCERRIM	Error injection mask for the ECC bits. A set bit corresponding to an ECC bit causes that bit to be inverted on SRAM writes if DERRIEN = 1.

7.3.1.5.2 Error Control and Capture Registers

The error control and capture registers control detection and reporting of tag parity, ECC and L2 configuration errors. L2 configuration errors are illegal combinations of L2 size and block size and

are detected when the L2 is enabled (L2CTL[L2E] = 1). Figure 7-14 shows the L2 error capture data high register (L2CAPTDATAHI).

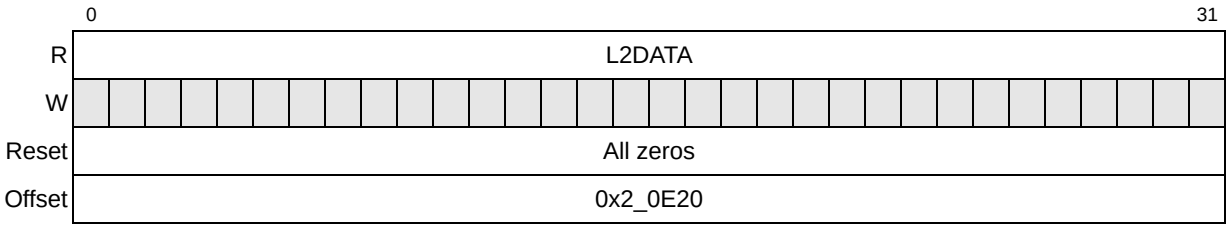


Figure 7-14. L2 Error Capture Data High Register (L2CAPTDATAHI)

Table 7-9 describes L2CAPTDATAHI[L2DATA] fields.

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).

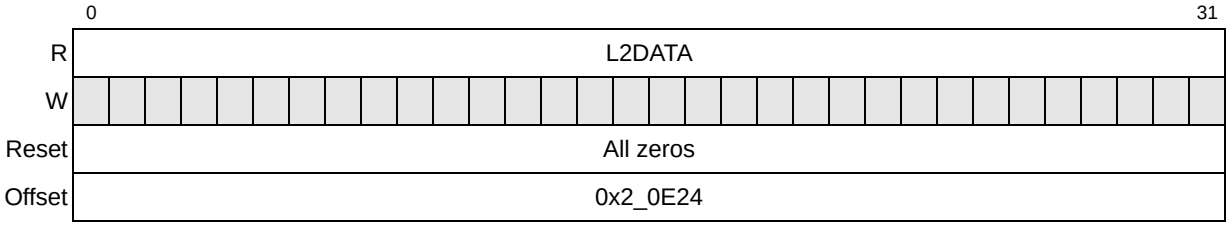


Figure 7-15. L2 Error Capture Data Low Register (L2CAPTDATALO)

Table 7-10 describes L2CAPTDATALO[L2DATA] fields.

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).

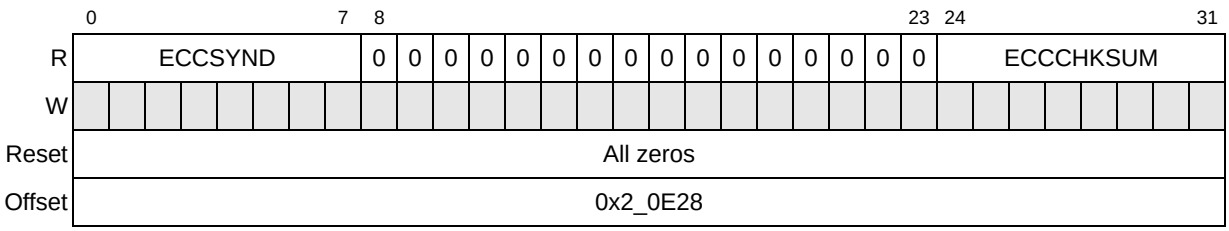


Figure 7-16. L2 Error Syndrome Register (L2CAPTECC)

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 double word
8–23	—	Reserved
24–31	ECCCHKSUM	The data path ECC of the failing double word

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

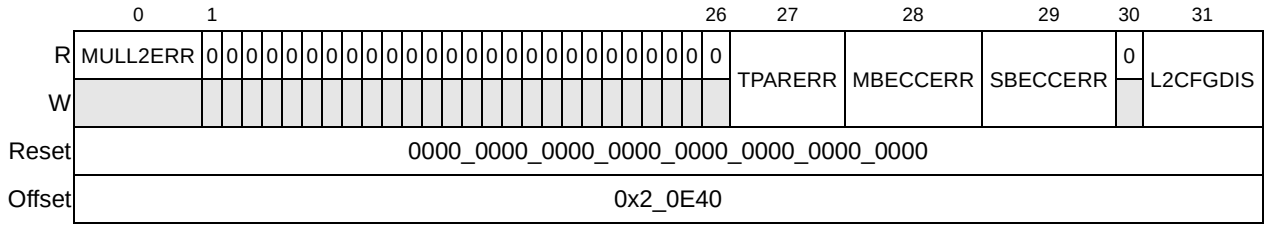


Figure 7-17. L2 Error Detect Register (L2ERRDET)

Table 7-12 describes L2ERRDET fields.

Table 7-12. L2ERRDET Field Descriptions

Bits	Name	Description
0	MULL2ERR	Multiple L2 errors (bit reset, write 1 to clear) 0 Multiple L2 errors of the same type were not detected. 1 Multiple L2 errors of the same type were detected.
1–26	—	Reserved
27	TPARERR	Tag parity error (bit reset, write 1 to clear) 0 Tag parity error was not detected 1 Tag parity error was detected Note that if an L2 cache tag parity error occurs on an attempt to write a new line, the L2 cache must be flash invalidated. L2 functionality is not guaranteed if flash invalidation is not performed after a tag parity error.
28	MBECCERR	Multiple-bit ECC error (bit reset, write 1 to clear) 0 Multiple-bit ECC errors were not detected. 1 Multiple-bit ECC errors were detected.
29	SBECCERR	Single-bit ECC error (bit reset, write 1 to clear) 0 Single-bit ECC error was not detected 1 Single-bit ECC error was detected
30	—	Reserved
31	L2CFGERR	L2 configuration error (bit reset, write 1 to clear). Reports inconsistencies between the L2SIZ, L2BLKSZ, and L2SRAM settings of the L2 control register (L2CTL). 0 L2 configuration errors were not detected 1 L2 illegal configuration error detected

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

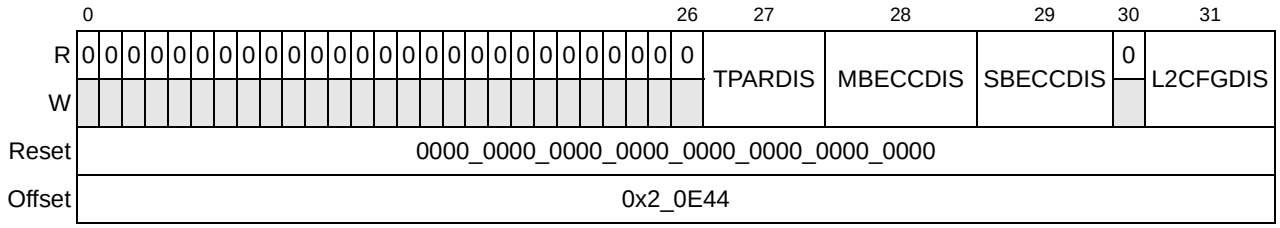


Figure 7-18. L2 Error Disable Register (L2ERRDIS)

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. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, MBECCDIS must be cleared and L2ERRINTEN[MBECCINTEN] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 0 Multiple-bit ECC error detection enabled 1 Multiple-bit ECC error detection disabled
29	SBECCDIS	Single-bit ECC error disable 0 Single-bit ECC error detection enabled 1 Single-bit ECC error detection disabled
30	—	Reserved
31	L2CFGDIS	L2 configuration error disable 0 L2 configuration error detection enabled 1 L2 configuration error detection disabled

Figure 7-19 shows the L2 error interrupt enable register (L2ERRINTEN). When an enabled error condition exists, the L2 signals an interrupt to the core through the internal *int* signal.

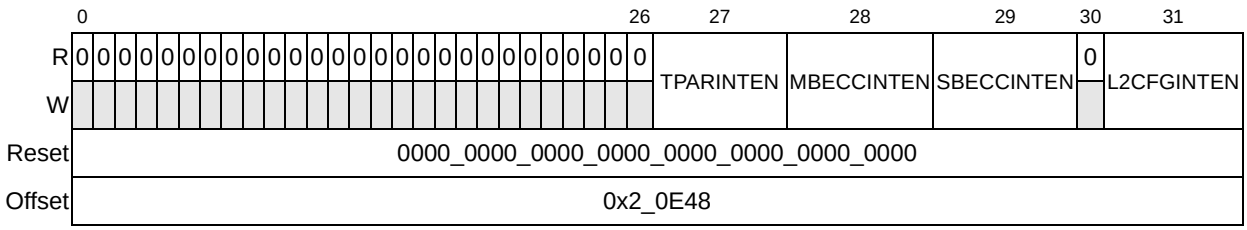


Figure 7-19. L2 Error Interrupt Enable Register (L2ERRINTEN)

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. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, L2ERRDIS[MBECCDIS] must be cleared and MBECCINTEN must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 0 Multiple-bit ECC error reporting disabled 1 Multiple-bit ECC error reporting enabled
29	SBECCINTEN	Single-bit ECC error reporting enable 0 Single-bit ECC error reporting disabled 1 Single-bit ECC error reporting enabled
30	—	Reserved
31	L2CFGINTEN	L2 configuration error reporting enable 0 L2 configuration error reporting disabled 1 L2 configuration error reporting enabled

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

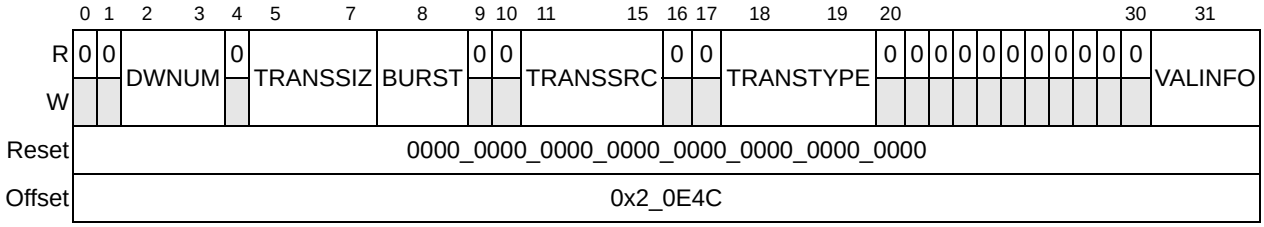


Figure 7-20. L2 Error Attributes Capture Register (L2ERRATTR)

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

Table 7-15. L2ERRATTR Field Descriptions (continued)

Bits	Name	Description
5–7	TRANSSIZ	Transaction size for detected error Single-beat / Burst / Single-beat / Burst 000 8 bytes — Reserved — 100 4 bytes — Reserved 001 1 byte — 16 bytes — 101 5 bytes — Reserved 010 2 bytes — 32 bytes — 110 6 bytes — Reserved 011 3 bytes — Reserved — 111 7 bytes — Reserved
8	BURST	Burst transaction for detected error 0 Single-beat (≤ 64 bits) transaction 1 Burst transaction
9–10	—	Reserved
11–15	TRANSSRC	Transaction source for detected error 00000 External (system logic) 10000 Processor (instruction) 10001 Processor (data)
16–17	—	Reserved
18–19	TRANSTYPE	Transaction type for detected error 00 Snoop (tag/status read) 01 Write 10 Read 11 Read-modify-write
20–30	—	Reserved
31	VALINFO	L2 capture registers valid 0 L2 capture registers contain no valid information or no enabled errors were detected. 1 L2 capture registers contain information of the first detected error which has reporting enabled. Software must clear this bit to unfreeze error capture so error detection hardware can overwrite the capture address/data/attributes for a newly detected error.

Figure 7-21 shows the L2 error address capture register (L2ERRADDR).

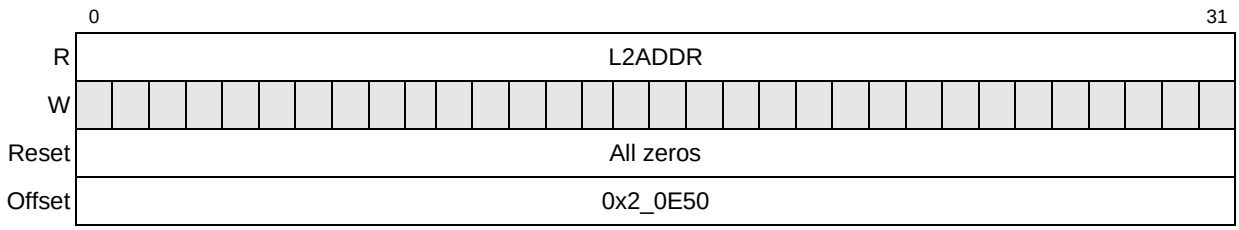


Figure 7-21. L2 Error Address Capture Register (L2ERRADDR)

Table 7-16 describes L2ERRADDR[L2ADDR] fields.

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).

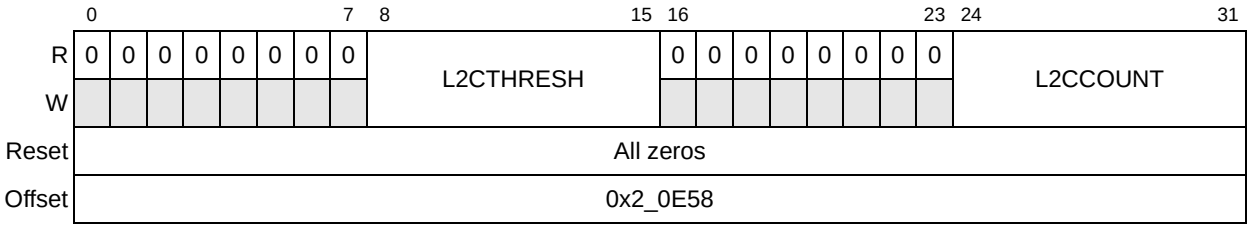


Figure 7-22. L2 Error Control Register (L2ERRCTL)

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
16–23	—	Reserved
24–31	L2CCOUNT	L2 count. Counts ECC single-bit errors detected. If L2CCOUNT equals the ECC single-bit error trigger threshold, an error is reported if single-bit error reporting is enabled

7.4 External Writes to the L2 Cache (Cache Stashing)

Data from an I/O master can be allocated into the L2 cache while simultaneously being written to memory. External writes can be performed from any I/O master:

- Ethernet
- CPM
- RapidIO
- PCI
- DMA

New cache lines are allocated for full cache line writes, unless the line is already valid or locked. Sub–cache-line write data is stashed only if the line is valid in the cache. A read-modify-write process is used to merge the stashed data with the valid line data.

The L2 cache external write address registers 0–3 (L2CEWAR*n*) are paired with the L2 cache external write control registers 0–3 (L2CEWCR*n*) to control the cache stashing functionality. Each register pair (for example, L2CEWAR0 and L2CEWCR0) specifies a programmed memory range that can be locked with a snoop write transaction. The address register must be naturally aligned to the window size in the corresponding control register. For more information, see Section 7.3.1.2, “L2 Cache External Write Address Registers 0–3 (L2CEWARn),” and Section 7.3.1.3, “L2 Cache External Write Control Registers 0–3 (L2CEWCRn).”

Note that stashing can occur regardless of whether the L1 cache is enabled.

Note that stashing can also occur even if the cache-inhibited bit in the MMU is set for the page. When the core looks for (and does not find) the stashed data in the L1 cache, a transaction is generated in which the L2 cache responds by updating the L1 cache with the requested data.

For information on how to initiate cache stashing, see the respective chapters for the I/O masters, listed above, that support stashing.

7.5 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 L2 cache frequency. The L2 returns the 128 bits containing the requested data (critical quad word) first. This data is forwarded to the result register before the full cache line reloads the L1.

Table 7-18. Fastest Read Timing—Hit in L2

Core Clocks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
e500 core load 1	To D-cache	D-cache miss	To CIU	CIU Q												To CIU	LSU DLFB	LSU reads command	LSU reads out data	Result bus						
e500 core load 2		to D-cache	D-cache miss	to CIU	CIU Q																To CIU	LSU DLFB	LSU reads command	LSU reads out data	Result bus	
CCB clocks	<1	2		3		4		5		6		7		8		9		10		11>						
CCB address bus load 1		BG		TS				AACK		HIT DATA-COMING																
CCB address bus load 2						BG		TS				AACK		HIT DATA-COMING												
CCB data bus load 1												DATA		DATA												
CCB data bus load 2																DATA		DATA								

7.6 L2 Cache and SRAM Coherency

This section explains the rules of cache and memory-mapped SRAM coherency. The term ‘snoop transaction’ refers to transactions initiated by the MPC8560 system logic or by I/O traffic, as opposed to e500 core-initiated transactions.

7.6.1 L2 Cache Coherency Rules

L2 cache coherency rules are as follows:

- The L2 is non-inclusive of the L1—valid L1 lines may be valid or invalid in the L2.
- The L2 cache holds no modified data. There are four states—invalid, exclusive, exclusive locked, and stale.
- The L2 allocates entries for data cast-out or pushed (non-global, non-write-through write with kill) from the L1 caches.
- Lines for e500 core-initiated burst read transactions are allocated as exclusive in the L2.
- The L2 supports I/O devices reading data from valid lines in the L2 cache (data intervention) if L2CTL[L2INTDIS] = 0. An optional unlock attribute causes I/O reads to clear a lock when the read is performed.
- The L2 cache does not respond to cache-inhibited read transactions.
- e500 core-initiated, cache-inhibited store transactions invalidate the line when they hit on a valid L2 line. If the line is locked, it goes to the stale state. For other write transactions the cache-inhibited bit is ignored.
- Non-burst cacheable write transactions from the e500 core (generated by write-through cacheable stores) update a valid L2 cache line through a read-modify-write operation.
- e500 core cast out transactions that hit on a stale line in the L2 cache cause a data update of the line and a change to the valid locked state for that line.
- An e500 core-initiated, cacheable, non-write-through store that misses in L1 and hits on a line in the L2 invalidates that line in the L2. If the line is marked exclusive locked, the L2 marks the line as stale.
- Transactions that hit a stale L2 cache line that would cause an allocate if they miss cause a data update of the line (when data arrives from memory) and a change to the line’s valid locked state. Data is not supplied by the L2 cache for the read in this case.
- The following transactions kill the data and the respective locks when they hit a valid L2 line:
  - **dcbf**
  - **dcbi**
- The L2 cache supports mixed cache external writes and core-initiated writes to the same addresses if the core-initiated writes are marked coherency-required, caching allowed, not

write-through (WIMG = 001x) and the external writes are marked coherency-required, caching-allowed.

- The L2 cache supports writes to the L2 cache from peripheral devices or from I/O controllers through snoop write transactions with addresses that hit in a programmed memory range. Full-cache-line (32-byte) write transactions update the data for a valid line in the L2, and if the line is not valid in the L2, a line is allocated. Sub-cache-line write transactions update the data only for valid L2 cache lines through read-modify-write operations.
- The L2 cache supports burst writes that lock an L2 cache line from peripheral devices or from I/O controllers through write transactions with addresses that hit in a programmed memory range that has the lock attribute set.
- The L2 cache supports burst writes that allocate and/or lock an L2 cache line from peripheral devices or I/O controllers through a write allocate transaction. See the system logic programming model (for example, that of the DMA controller) for details on how to set the transaction type for cache external writes to the L2.

7.6.2 Memory-Mapped SRAM Coherency Rules

Memory-mapped SRAM coherency rules are as follows:

- External (non–core-initiated) accesses to memory-mapped SRAM must be marked coherency-required. External accesses marked coherency-not-required to memory-mapped SRAM may cause an address unavailable error.
- Accesses to memory-mapped SRAM are cacheable only in the corresponding e500 L1 caches. External accesses must be marked cache-inhibited or be performed with non-caching transactions.

7.7 L2 Cache Locking

The MPC8560 caches can be locked and cleared using the following methods:

- Cache locking methods
  - Individual line locks are set and cleared by using instructions defined by the e500 cache locking APU, which is part of the Freescale Book E Implementation Standards (EIS). These instructions include Data Cache Block Touch and Lock Set (**dcbtls**), Data Cache Block Touch for Store and Lock Set (**dcbtstls**), and Instruction Cache Block Touch and Lock Set (**icbtls**). For detailed information about these instructions, see the *PowerPC e500 Core Reference Manual.*
  - A lock attribute can be attached to write operations.

  - Individual line locks are set and cleared through core-initiated instructions, by external reads or writes, or by accesses to programmed memory ranges defined in L2 cache external write address registers (L2CEWARn).
  - The entire cache can be locked by setting configuration registers appropriately.
- Methods for clearing locks
  - Individual locks are cleared by cache locking APU instructions (Instruction Cache Block Lock Clear (**icblc**) and Data Cache Block Lock Clear (**dcblc**)) or by snooped flush unless the entire cache is locked.
  - Flash clearing of all instruction and/or data locks can be done by writes to configuration registers.
  - An unlock attribute can be attached to I/O read operations.

7.7.1 Locking the Entire L2 Cache

The entire L2 cache can be locked by setting L2CTL[L2DO] = 1 and L2CTL[L2IO] = 1. This has the effect of preventing any further allocation of new lines in the cache by core requests. If there are lines in the cache that are not valid, they cannot be used by core requests until the cache is unlocked. While the cache is locked, read requests are serviced as normal, and snooping continues as normal to maintain coherency. Lines invalidated to satisfy coherency requirements cannot be reallocated by core requests while the cache remains locked. The L2 cache can be unlocked by clearing L2CTL[L2IO] and/or L2CTL[L2DO]. Note that L2CTL[L2DO] and L2CTL[L2IO] have no effect on cache external write allocations or memory-mapped SRAM.

Note that this form of cache locking does not use the lock bits of the cache and cannot be cleared by resetting the cache or lock bits.

7.7.2 Locking Programmed Memory Ranges

A programmed memory range can be locked with a snoop write transaction that matches a cache external write address range (specified by L2CEWAR*n* and L2CEWCR*n*). There is no clearing of locks through the programmed address ranges. Locks can be cleared using clear lock instructions, flushes, or read-and-clear-lock snoop (RWNITC with clear lock attribute), or flash clear locks.

7.7.3 Locking Selected Lines

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

- **icbtls** (CT = 1)—Instruction Cache Block Touch and Lock Set instruction
- **dcbtls** (CT = 1)—Data Cache Block Touch and Lock Set instruction
- **dcbtstls** (CT = 1)—Data Cache Block Touch for Store and Lock Set instruction

- Snoop burst write—If the address hits on a programmed cache external write space with the lock attribute set, or if the write allocate transaction type is used
- Snoop non-burst write—If the address hits on a programmed cache external write space with the lock attribute set

Note that the core complex broadcasts these instructions to the L2 if the CT field in the instruction specifies the L2 cache (CT = 1). When the L2 cache is specified, data is not placed in the L1, only the L2. If the L1 cache is specified (CT = 0), the L2 does not lock the line, and the data is placed in the L1 (and locked).

When the touch lock set L2 instruction (**dcbtls** or **dcbtstls**) hits are modified in the L1 cache, the modified data is allocated into the L2 cache (and written back to main memory) and a data lock is set. The L1 line state transitions to invalid.

Note that if the L2 receives a request to allocate and lock a line, but all lines in the selected way are locked, the requested L2 line is not allocated and the L2 cache lock overflow bit (L2CTL[L2LO]) is set.

Lines invalidated to satisfy coherency requirements cannot be reallocated while the cache remains locked.

7.7.4 Clearing Locks on Selected Lines

Individual locks in the L2 are cleared by a lock clear (**icblc** or **dcblc,** CT = 1) instruction. This directs the L2 cache to clear a lock on that line if it hits in the L2 cache. Both data and instruction locks are cleared by the **icblc** and **dcblc** instructions.

Note that the lock on a line is cleared if the line is invalidated by a snooped flush transaction, and the line in the cache is available for allocation of a new line of instruction or data unless the entire cache is locked.

NOTE

There is a scenario in which a lock clear operation appears to fail to clear a lock in the L2 cache. This occurs only when the attempt to set the lock results in a bus error (for example, PCI returns an error condition).

Assume the following scenario:

1. The e500 attempts to set a lock in the L2 cache (by executing a **dcbtls** or **icbtls** instruction with CT = 1). The line is not already present in the cache, so it must be read from external memory. This read encounters an error which, depending on the chip configuration, will be reported to the core (probably as an interrupt).

2. At (or near) the same time, a cache external write to the same cache line is being mastered by the ECM.

3. Very soon after the cache external write, a transaction to clear the lock occurs. This can be caused by the processor executing a **dcblc** or **icblc** instruction with CT = 1, or by the ECM mastering a lock clear transaction.

If this scenario occurs within a tight timing window, the cache line may unexpectedly remain locked at the end of the sequence.

The interrupt handler may want to clear the erroneously remaining lock in this case.

7.7.5 Flash Clearing of Instruction and Data Locks

Locks for instructions and data are recorded separately in the L2 cache, and they can be flash cleared separately by writing the appropriate value to the L2 cache control register (L2CTL[L2LFR] and L2CTL[L2LFRID]). Flash invalidating of the L2 (setting L2CTL[L2I]) clears all locks on both instructions and data.

Note that flash clearing is the only way to clear data locks without clearing instruction locks, or to clear instruction locks without clearing data locks. All instructions and snoop transactions that clear locks clear both data and instruction locks.

7.7.6 Locks with Stale Data

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

7.8 PLRU L2 Replacement Policy

Line replacement is determined using a pseudo least-recently-used (PLRU) algorithm. There is a valid bit (V0–V7) for each line. To determine the replacement victim (the line to be cast out), there are seven PLRU bits (P0–P6) for each set. PLRU bits are updated every time a new line is allocated or replaced and every time a line is modified or invalidated. 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. The eight ways of the L2 cache are labeled W0–W7; the seven PLRU bits are labeled P0–P6.

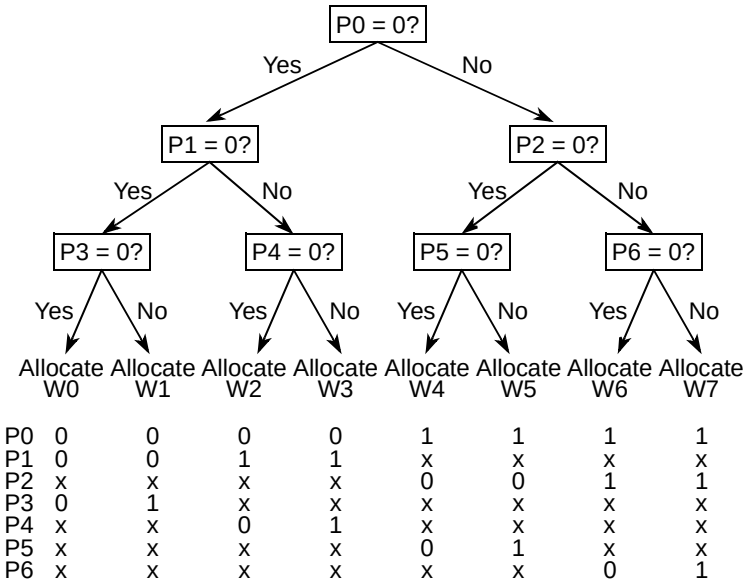


Figure 7-23. L2 Cache Line Replacement Algorithm

7.8.1 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 last accessed, as summarized in Table 7-19.

Table 7-19. PLRU Bit Update Algorithm

Last Way Accessed	PLRU Bits						
	P0	P1	P2	P3	P4	P5	P6
0	1	1	—	1	—	—	—
1	1	1	—	0	—	—	—
2	1	0	—	—	1	—	—
3	1	0	—	—	0	—	—
4	0	—	1	—	—	1	—
5	0	—	1	—	—	0	—
6	0	—	0	—	—	—	1
7	0	—	0	—	—	—	0

When an L2 line is invalidated, the PLRU bits are updated, marking the corresponding way as least-recently-used. This causes the invalidated way to be selected as the next victim.

7.8.2 Allocation of Lines

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

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

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

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

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

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

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

7.9 L2 Cache Operation

This section describes the behavior of the L1 and L2 cache in response to various operations and in various configurations.

7.9.1 L2 Cache States

The L2 status array uses four bits for each line to determine the status of the line. Different combinations of these bits result in different L2 states. The status bits are as follows:

- Valid (V)
- Instruction locked (IL)
- Data locked (DL)
- Stale (T)

Table 7-21 shows L2 cache states. Note that these conventions are also used in Table 7-22.

Table 7-21. L2 Cache States

V	T	IL	DL	L2 states
0	x	x	x	Invalid (I)
1	0	0	0	Exclusive (E)
1	0	0	1	Exclusive data locked (EDL)
1	0	1	0	Exclusive instruction locked (EIL)
1	0	1	1	Exclusive instruction and data locked (EL)
1	1	0	0	Stale (data invalid, locks invalid) (T)
1	1	0	1	Stale (data invalid, dlock valid) (TDL)
1	1	1	0	Stale (data invalid, ilock valid) (TIL)
1	1	1	1	Stale (data invalid, locks valid) (TL)

7.9.2 Flash Invalidation of the L2 Cache

The L2 cache may be completely invalidated by setting the L2I bit of the L2 control register (L2CTL). Note that no data is lost in this process because the L2 cache is a write-through cache and contains no modified data. Flash invalidation of the cache is necessary when the cache is initially enabled and may be necessary to recover from some error conditions such as a tag parity error.

The invalidation process requires several cycles to complete. The L2I bit remains set during this procedure and is then cleared automatically when the procedure is complete. The L2 cache controller issues retries for all transactions on the e500 core complex bus (CCB) while the flash invalidation process is in progress.

Note that the contents of memory-mapped SRAM regions of the data array are unaffected by a flash invalidation of the L2 cache regions fo the array.

7.9.3 L2 State Transitions

Table 7-22 lists state transitions for all e500 core-initiated transactions that change the L2 cache state. Core-initiated transactions caused when the core executes **msync**, **mbar**, **tlbivax**, or **tlbsync** do not change the L2 cache state. The table does not list initial L1 states for transactions that hit in the L1 (iL1 or dL1) and are not sent to the L2.

In the table, the heading ‘L2 hit’ indicates that the L2 provides (on a read) or captures (on a write) data for an existing line. Some entries list two final L1 states. L2 touch instructions never allocate into iL1 or dL1.

Note that if the L2 SRAM is disabled, the L2 initial and final states are always I and the L2 never hits. Similarly, if the L2 SRAM is in full memory-mapped SRAM mode, the L2 initial and final states are always I and the L2 never hits for addresses not in the memory-mapped SRAM address range. The L2 always hits for addresses in the enabled memory-mapped SRAM address ranges.

Table 7-22. State Transitions Due to Core-Initiated Transactions

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
Cacheable instruction fetch icbtls_L1	iL1 I	I/T	No	I/V	Same	L2CTL[L2DO] = 1. L2 touch instructions not allocated in L1
		I	No	I/V	E	L2CTL[L2DO] = 0
icbt_L2	dL1 I,E	E/EL	Yes	I/V	Same	
		T	No	I/V	EL	L2CTL[L2DO] = 0. Restore locked line in L2 with valid data from bus
icbtls_L2	dL1 I,E	I/T	No	I	Same	L2CTL[L2DO] = 1
		E	Yes	I	I	L2CTL[L2DO] = 1
		EL	Yes	I	T	L2CTL[L2DO] = 1
		I	No	I	EL	L2CTL[L2DO] = 0
		E	Yes	I	EL	L2CTL[L2DO] = 0
		EL	Yes	I	Same	L2CTL[L2DO] = 0
		T	No	I	EL	L2CTL[L2DO] = 0. Restore locked line in L2 with valid data from bus
Cache-inhibited instruction fetch	N/A	N/A	No	N/A	N/A	No L1/L2 effect
Cacheable load (4-state) Cacheable **lwarx** (4-state) dcbt_L1 (4-state) dcbtls_L1 (4-state)	dL1 I	I/T	No	E	Same	L2CTL[L2IO] = 1
		E	Yes	E	I	L2CTL[L2IO] = 1
		EL	Yes	E	T	L2CTL[L2IO] = 1
		I	No	E	E	L2CTL[L2IO] = 0
		E/EL	Yes	E	Same	L2CTL[L2IO] = 0
		T	No	EL	EL	L2CTL[L2IO] = 0. Restore locked line in L2 with valid data from bus
Cache-inhibited load	N/A	N/A	No	N/A	N/A	No L1/L2 effect
Cache-inhibited **lwarx**	N/A	N/A	No	N/A	N/A	No L2 effect
Writeback Store	dL1 I	I/T	No	M	Same	L2 allocates when a line is cast out of L1.
		E	Yes	M	I	
		EL	Yes	M	T	
Writeback **stwcx.**	dL1 I	I/T	No	M	Same	
		E	Yes	M	I	
		EL	Yes	M	T	

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

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
Cacheable load (3-state) Cacheable **lwarx** (3-state) dcbt_L1 (3-state) dcbtls_L1 (3-state)	dL1 I	I	No	E/I	I	L2CTL[L2IO] = 1
		T	No	E/I	T	L2CTL[L2IO] = 1
		E	Yes	E/I	I	L2CTL[L2IO] = 1
		EL	Yes	E/I	T	L2CTL[L2IO] = 1
		I	No	E/I	E	L2CTL[L2IO] = 0
dcbt_L2 dcbtst_L2	dL1 I,E	E/EL	Yes	E/I	Same	L2CTL[L2IO] = 0
		T	No	E/I	EL	L2CTL[L2IO] = 0. Restore locked line with valid data from bus
dcbtst_L1 dcbtstls_L1	dL1 I	I/T	No	E	Same	
		E	Yes	E	I	
		EL	Yes	E	T	
dcbtls_L2 dcbtstls_L2	dL1 I,E	I	No	I	I	L2CTL[L2IO] = 1
		T	No	I	T	L2CTL[L2IO] = 1
		E	Yes	I	I	L2CTL[L2IO] = 1
		EL	Yes	I	T	L2CTL[L2IO] = 1
		I	No	I	EL	L2CTL[L2IO] = 0
		E/EL	Yes	I	EL	L2CTL[L2IO] = 0
		T	No	I	EL	L2CTL[L2IO] = 0. Restore locked line with valid data from bus
Write-through store	dL1 I,E,M	I/T	No	Same	I	
		E/EL	Yes	Same	Same	Read-modify-write
Cache-inhibited store	N/A	I/E	No	N/A	I	Invalidate line
		EL/T	No	N/A	T	Invalidate data, keep lock
Cache-inhibited **stwcx.**	N/A	I/E	No	N/A	I	Invalidate line
		EL/T	No	N/A	T	Invalidate data, keep lock
dcblc_L2 icblc_L2	dL1 I,E,M	I/E	No	Same	Same	
		EL	No	Same	E	
		T	No	Same	I	
Victim castout dcbt_L2 icbt_L2 dcbtst_L2	dL1 M	I/T	No	I	Same	L2CTL[L2IO] = 1. If software sharing cache lines between instructions and data wishes to capture instruction lines in L2 with L2CTL[L2IO] = 1, it must perform **dcbst** to flush the line out of the dL1 before fetching it into L2.
		I	No	I	E	L2CTL[L2IO] = 0
		E/EL	No	I	I/T	L2CTL[L2IO] = 1
			Yes	I	Same	L2CTL[L2IO] = 0
		T	Yes	I	EL	L2CTL[L2IO = 0

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

Source of Transaction	Initial States		L2 Hit	Final States		Comments
	L1	L2		L1	L2	
dcbtls_L2 icbtls_L2 dcbtstls_L2	dL1 M	I	No	I	EL	An icbtls_L2 that hits modified in L1 cannot be distinguished from dcbtls_L2 and sets the L2 dlock bit. If software shares cache lines between instructions and data and wishes to set ilocks in L2, it must perform **dcbst** to flush the line out of the dL1 before locking it in L2.
		E/EL/T	Yes	I	EL	
Snoop push	dL1 M	I/E	No	I/E	I	
		EL/T	No	I/E	T	Invalidate data, keep lock
dcbf **dcbst**	dL1 M	I/E/EL	No	I	I	
dcbz **dcba**	dL1 I	I/E	No	M	I	
		EL	No	M	T	
dcbi	dL1 I,E,M	I/ E/EL/T	No	I	I	
dcbf **dcbst**	dL1 I,E	I/ E/EL/T	No	I	I	
icbi	iL1 I,V	I/ E/EL/T	No	I	I	

Table 7-23 lists L2 cache state transitions for all system-initiated (non-core) 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 7-23 accounts for changes caused by L1 snoop pushes triggered by snoops, listed in Table 7-22.

Table 7-23. 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

Table 7-23. State Transitions Due to System-Initiated Transactions (continued)

Transaction Type	$\overline{wt}$	$\overline{ci}$	$\overline{gbl}$	Initial L2 State	Final L2 State	Comments
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
< 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/L/T	Same	
				E	E	
				EL	EL	
	0	1	0	I	Same	Read-and-clear-lock
				EL	E	
				T	I	
	x	0	0	N/A	N/A	No L1/L2 effect
Kill RWITM RWITM atomic RClaim	x	1	0	I/E	I	
				EL/T	T	Invalidate data, keep lock
	x	0	0	I/E/EL/T	Same	

7.10 Initialization/Application Information

This section describes some required steps for initializing the L2 SRAM both in L2 cache mode, and in memory-mapped SRAM mode. Also, it includes some guidlines for error managment.

7.10.1 Initialization

This section describes L1 cache and memory-mapped SRAM initialization.

7.10.1.1 L2 Cache Initialization

After power-on reset, the valid bits in the L2 cache status array are in random states. Therefore, it is necessary to perform a flash invalidate operation before using the array as an L2 cache. This is done by writing a 1 to the L2I bit of the L2 control register (L2CTL). This can be done before or simultaneously with the write that enables the L2 cache. That is, the L2E and L2I bits of L2CTL can be set simultaneously. The L2I bit clears automatically, so no further writes are necessary.

7.10.1.2 Memory-Mapped SRAM Initialization

After power-on reset, the contents of the data and ECC arrays and are random, so all SRAM data must be initialized before it is read. If the cache is initialized by the core or any other device that uses sub-cacheline transactions, ECC error checking should be disabled during the initialization process to avoid false ECC errors generated during the read-modify-write process used for sub-cacheline writes to the SRAM array. This is done by setting the multi- and single-bit ECC error disable bits of the L2 error disable register (L2ERRDIS[MBECCDIS, SBECCDIS]). See Section 7.3.1.5.2, “Error Control and Capture Registers”. If the array is initialized by a DMA engine using cache-line writes, then ECC checking can remain enabled during the initialization process.

7.10.2 Managing Errors

This section describes recommended handling for ECC and tag parity errors.

7.10.2.1 ECC Errors

An individual soft error that causes a single- or mulit-bit ECC error can be cleared from the L2 array simply by executing a **dcbf** instruction for the address captured in the L2ERRADDR register. This will invalidate the line in the L2 cache. When the load that caused the ECC error is performed again, the data will be re-allocated into the L2 with ECC bits set properly again.

If the threshold for single bit errors set in the L2ERRCTL register is exceeded, then the L2 cache should be flash invalidated to clear out all single-bit errors.

Note that no data is lost by executing **dcbf** instructions or flash invalidate operations because the L2 cache is write-through and contains no modified data.

7.10.2.2 Tag Parity Errors

A tag parity error must be fixed by flash invalidating the L2 cache. Note that executing a **dcbf** instruction for the address that caused the error to be reported is not sufficient because a tag parity error is seen as an L2 miss and does not cause invalidation of the bad tag. Proper L2 operation cannot be guaranteed if an L2 tag parity error is not repaired by a flash invalidation of the entire array.

Part III
Memory and I/O Interfaces

Part III 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. The following chapters are included:

- 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 override, crawl mode, and ECC error injection support rapid system debug.
- Chapter 10, "Programmable Interrupt Controller," describes the embedded programmable interrupt controller (PIC) of the 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, "I^2C 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 Controllers,” describes the two three-speed Ethernet controllers on the MPC8560. These controllers provide 10/100/1Gb 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/PCI-X 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

The e500 coherency module (ECM) provides a flexible, easily expandable switching structure for routing e500- and I/O-initiated transactions to target modules on the device. Figure 8-1 shows a high-level block diagram of the ECM.

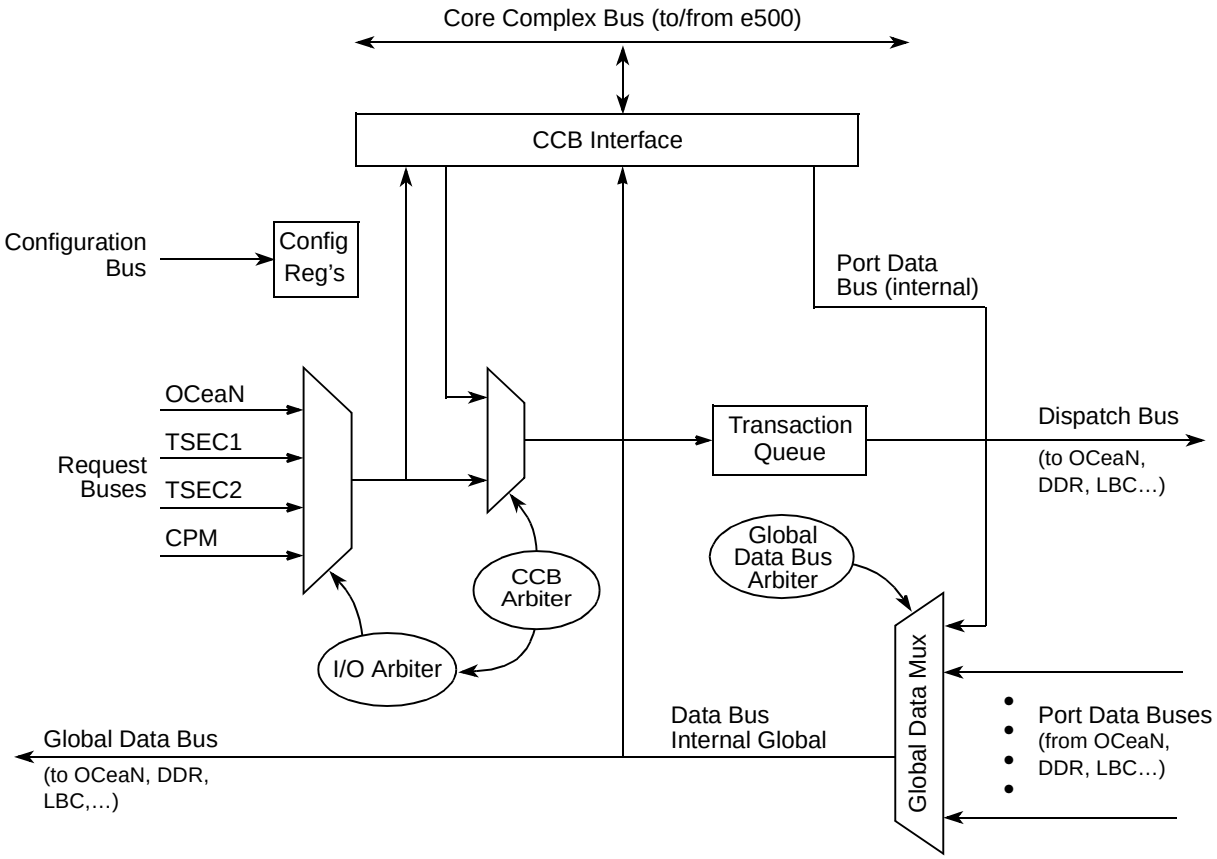


Figure 8-1. e500 Coherency Module Block Diagram

8.1.1 Overview

The ECM routes transactions initiated by the e500 core to the appropriate target interface on the device. In a manner analogous to a bridging router in a local area network, the ECM forwards I/O-initiated transactions that are tagged with the global attribute onto the core complex bus

(CCB). This allows on-chip caches to snoop these transactions as if they were locally initiated and to take actions to maintain coherency across cacheable memory.

8.1.2 Features

The ECM includes these distinctive features:

- Support for e500 core and an L2/SRAM on the CCB, including a CCB arbiter. It sources a 64-bit data bus for returning read data from the ECM to the e500 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 and a 128-bit write data bus from the e500 core.
- Four connection points for I/O initiating (mastering into the device) interfaces. One of those connection points services OCeaN initiators, one services TSEC1, one services TSEC2, 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 (PIC), and configuration register access block all have a target port connection to the ECM.
- Split transaction support—separate address and data tenures allow for pipelining of transactions and out-of-order data tenures between initiators and targets.
- Proper ordering of I/O-initiated transactions.
- Speculative read bus for low-latency dispatch of reads to the DDR controller.
- Low-latency path for returning read data from DDR to the e500.
- Error registers trap transactions with invalid addresses. Errors can be programmed to generate interrupts to the e500 core, as described in the following sections:
  - Section 8.2.1.3, “ECM Error Detect Register (EEDR)”
  - Section 8.2.1.4, “ECM Error Enable Register (EEER)”
  - Section 8.2.1.5, “ECM Error Attributes Capture Register (EEATR)”
  - Section 8.2.1.6, “ECM Error Address Capture Register (EEADR)”
- Errors from I/O devices on reads (for example, a master-aborted read transaction on the PCI interface) terminate with corrupt data sent to the requester. If the requester is the e500 core, the ECM also asserts *core_fault_in* to the core, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and one of these errors occurs, appropriate interrupts must be enabled to ensure that an interrupt is generated. See Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

8.2 Memory Map/Register Definition

Table 8-1 shows the ECM's memory map. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 8-1. ECM Memory Map

Local Memory Offset	Register	Access	Reset	Section/Page
0x0_1000	EEBACR—ECM CCB address configuration register	R/W	0x0000_0003	8.2.1.1/8-3
0x0_1010	EEBPCR—ECM CCB port configuration register	R/W	0x0*n*00_0000	8.2.1.2/8-4
0x0_1E00	EEDR—ECM error detect register	Special	0x0000_0000	8.2.1.3/8-5
0x0_1E08	EEER—ECM error enable register	R/W	0x0000_0000	8.2.1.4/8-6
0x0_1E0C	EEATR—ECM error attributes capture register	R	0x0000_0000	8.2.1.5/8-7
0x0_1E10	EEADR—ECM error address capture register	R	0x0000_0000	8.2.1.6/8-8

8.2.1 Register Descriptions

This section consists of detailed descriptions of those registers summarized in Table 8-1. Note that these registers are shown in big-endian format.

8.2.1.1 ECM CCB Address Configuration Register (EEBACR)

The ECM CCB address configuration register, shown in Figure 8-2, controls arbitration and streaming policies for the CCB.

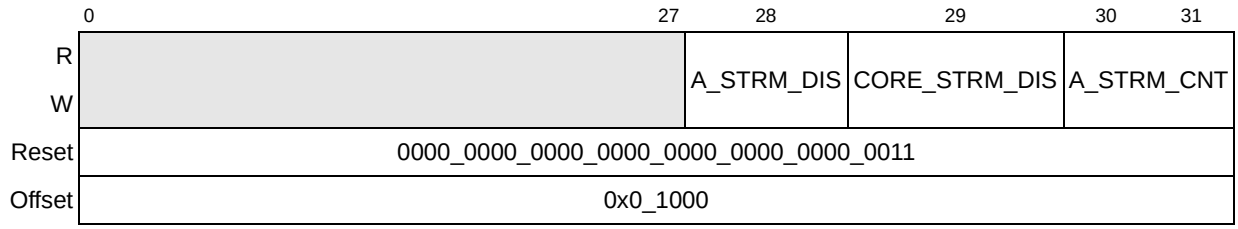


Figure 8-2. ECM CCB Address Configuration Register (EEBACR)

Table 8-2 describes the EEBACR fields.

Table 8-2. EEBACR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	A_STRM_DIS	Controls whether the ECM allows any streaming to occur. 0 Streaming is enabled. 1 Streaming is disabled.

Table 8-2. EEBACR Field Descriptions (continued)

Bits	Name	Description
29	CORE_STRM_DIS	With A_STRM_DIS, controls whether the e500 core can stream commands onto the CCB. A_STRM_DIS and CORE_STRM_DIS must both be cleared for the e500 core to be enabled to stream address tenures that it masters. 0 Stream address tenures initiated by the e500 core, provided A_STRM_DIS is cleared. 1 Streaming of address tenures initiated by the e500 core not allowed.
30–31	A_STRM_CNT	Stream count. Specifies the maximum number of transactions that any master can stream (issue sequentially without preemption) on the CCB following an initial transaction. 00 Reserved 01 One transaction can be streamed with the initial transaction. 10 Two transactions can be streamed with the initial transaction. 11 Three transactions can be streamed with the initial transaction. Default.

8.2.1.2 ECM CCB Port Configuration Register (EEBPCR)

The ECM CCB port configuration register (EEBPCR) is shown in Figure 8-3.

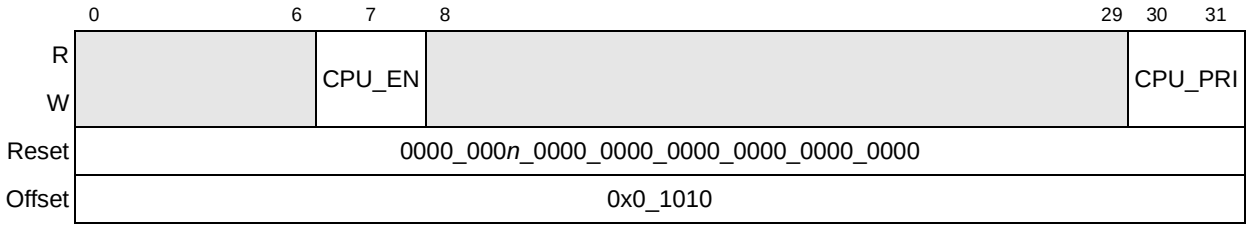


Figure 8-3. ECM CCB Port Configuration Register (EEBPCR)

Table 8-3 describes EEBPCR fields.

Table 8-3. EEBPCR Field Descriptions

Bits	Name	Description
0–6	—	Reserved
7	CPU_EN	CPU port enable. Controls boot holdoff mode when the device is an agent of an external host. Specifies whether the e500 core (CPU) port is enabled to run transactions on the CCB. The CPU boot configuration power-on reset pin (cfg_cpu_boot) determines the initial value of this bit. If the pin is sampled as a logic 1 at the negation of reset, the CPU is enabled to boot at the end of the POR sequence. Otherwise, the CPU cannot fetch its boot vector until an external host sets the CPU_EN bit. 0 Boot holdoff mode. CPU arbitration is disabled on the CCB and no bus grants are issued. 1 CPU is enabled and receives bus grants in response to bus requests for the boot vector. After this bit is set, it should not be cleared by software. It is not intended to dynamically enable and disable CPU operation. It is only intended to end boot holdoff mode. See Section 4.4.3.5, “CPU Boot Configuration,” for more information.

Table 8-3. EEBPCR Field Descriptions (continued)

Bits	Name	Description
8–29	—	Reserved
30–31	CPU_PRI	Specifies 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

8.2.1.3 ECM Error Detect Register (EEDR)

The ECM error detect register (EEDR) is shown in Figure 8-4.

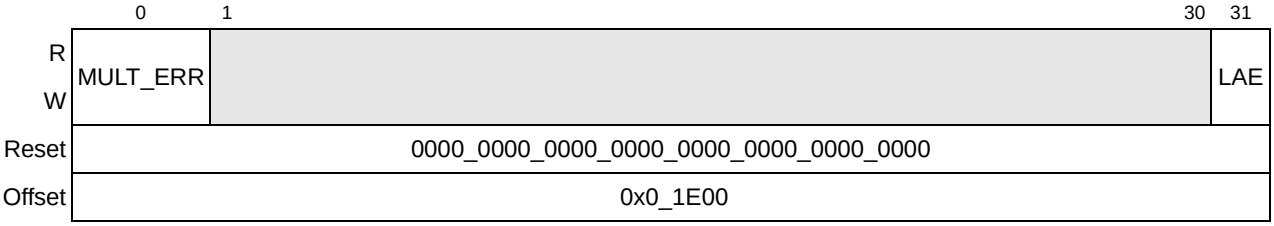


Figure 8-4. ECM Error Detect Register (EEDR)

Table 8-4 describes EEDR fields.

Table 8-4. EEDR Field Descriptions

Bits	Name	Description
0	MULT_ERR	Multiple error. Indicates the occurrence of multiple errors of the same type. Write one to clear. 0 Multiple errors of the same type were not detected. 1 Multiple errors of the same type were detected.
1–30	—	Reserved
31	LAE	Local access error. Write one to clear. Three cases can generate LAEs: • Transaction does not map to any target. In this case the ECM injects read responses (with the corrupt attribute set) and write data is dropped. Note that a read that attempts to access an unmapped target causes the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, EEER[LAEE] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).” • Source and target IDs indicate that an OCN port initiated a transaction that targets an OCN port. This loopback behavior can result from programming errors where inbound ATMU window targets are inconsistent with targets configured in the local access windows for a given address range. For this type of LAE, the dispatch (to OCN target in this case) is not screened off; the LAE error is reported, but the transaction is still sent to its OCN target. • Inbound read atomic set, clear, inc, or dec transactions from RapidIO target an interface other than DDR (only the DDR supports these transactions). This condition screens off the dispatch to the target (if one is mapped); the ECM injects read responses (with the corrupt attribute set) and write data is dropped. 0 Local access error has not occurred. 1 Local access error occurred.

8.2.1.4 ECM Error Enable Register (EEER)

The ECM error enable register (EEER) shown in Figure 8-5 enables the reporting of error conditions to the e500 core through the internal *int* interrupt signal.

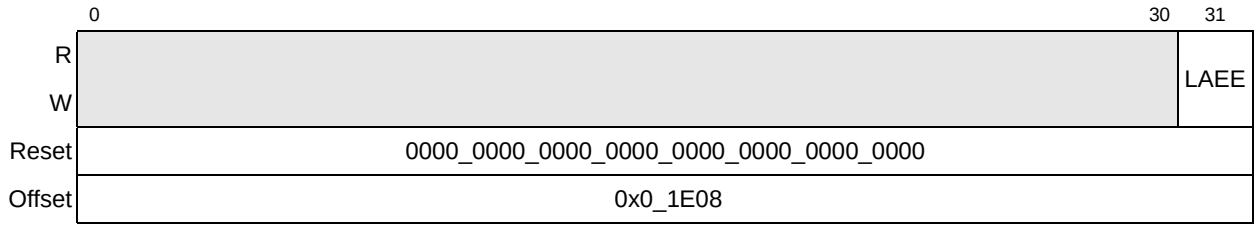


Figure 8-5. ECM Error Enable Register (EEER)

Table 8-5 describes EEER fields.

Table 8-5. EEER Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	LAEE	Local access error enable. Note that a read that attempts to access an unmapped target causes the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, LAEE must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 0 Disable reporting local access errors as interrupts. 1 Enable reporting local access errors as interrupts.

8.2.1.5 ECM Error Attributes Capture Register (EEATR)

The ECM error attributes capture register (EEATR) is shown in Figure 8-6.

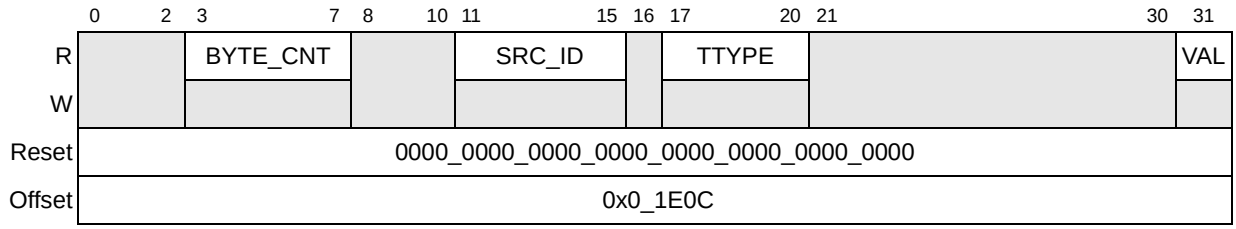


Figure 8-6. ECM Error Attributes Capture Register (EEATR)

Table 8-6 describes EEATR fields.

Table 8-6. EEATR Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3–7	BYTE_CNT	Byte count. Specifies the transaction byte count. 00000 32 bytes 00001 1 byte 00010 2 bytes 00011 3 bytes 00100 4 bytes 01000 8 bytes 10000 16 bytes
8–10	—	Reserved

Table 8-6. EEATR Field Descriptions (continued)

Bits	Name	Description
11–15	SRC_ID	Source ID. Specifies the source device mastering the transaction. 00000 PCI/PCI-X interface 00001–01001 Reserved 01010 Boot sequencer 01011 Reserved 01100 RapidIO 01101–01111 Reserved 10000 Processor (instruction) 10001 Processor (data) 10010 Reserved 10011 Reserved 10100 CPM 10101 DMA 10110 Reserved 10111 SAP 11000 TSEC1 11001 TSEC2 11010 Reserved 11011 Reserved 11100 RapidIO message units 11101 RapidIO doorbell units 11110 RapidIO port-write units 11111 Reserved
16	—	Reserved
17–20	TTYPE	Transaction type. Defined as follows: 0000 Write 0001 Reserved 0010 Write with allocate 0011 Write with allocate with lock 0100 Address only transaction 0101–0111 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.

8.2.1.6 ECM Error Address Capture Register (EEADR)

The ECM error address capture register (EEADR) is shown in Figure 8-7.

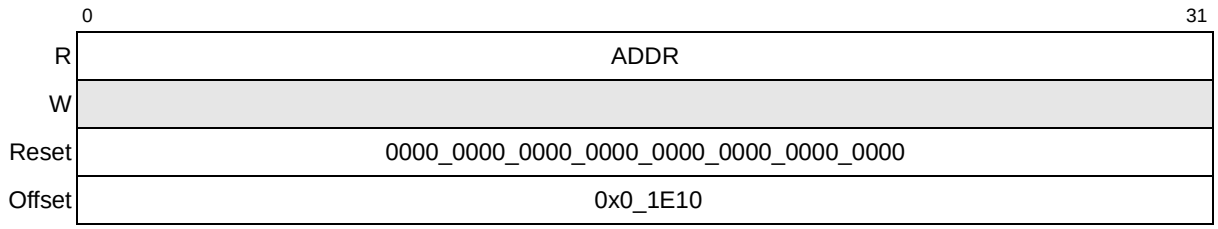


Figure 8-7. ECM Error Address Capture Register (EEADR)

Table 8-7 describes EEADR fields.

Table 8-7. EEADR Field Descriptions

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

8.3 Functional Description

The following is a very general discussion of ECM operation.

8.3.1 I/O Arbiter

Figure 8-1 shows the I/O arbiter block that manages I/O-initiated address tenure requests arriving on the request buses. Four request buses compete for access to the ECM, which can only process one request at a time. The ECM uses two factors to select the winning request bus: the primary factor is request priority and the secondary factor is longest waiting/least recently granted status. By default all requesters use the lowest priority (priority 0) for requests, except for the CPM, which always requests at the highest priority (priority 3). The TSEC controllers dynamically raise and lower their priority levels based on FIFO depth. A starvation avoidance algorithm prevents high priority requests from indefinitely starving low priority requesters. The transaction from the winning request bus competes with e500 core requests for the CCB and entry into the transaction queue.

8.3.2 CCB Arbiter

Figure 8-1 shows the CCB arbiter block coordinating the entry of new transactions into the ECM's transaction queue. It handles arbitration for requests to use the CCB from the e500 core and the winning request bus and consequently controls when these new transactions can enter the transaction queue.

Because the CCB bus operates most efficiently when it streams commands from one initiator, the CCB arbiter alternates grants between streams of transactions from the processor and from the winner of the I/O arbiter. The length of a stream (number of back-to-back transactions) is limited by the A_STRM_CNT field in the EEBACR register. However, the arbiter also uses the priority of the requests to limit streaming. If the priority of a new request is higher than that of a stream in progress, then the higher priority transaction will interrupt the other stream. The priority of e500 transactions is set by the CPU_PRI field in EEBPCR register.

8.3.3 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.

8.3.4 Global Data Multiplexor

Figure 8-1 shows how the global data multiplexor takes data bus connections and multiplexes them onto one 128-bit global data bus. The global data mux allows initiators of write transactions to route data to their targets and read targets to return data to the initiators.

8.3.5 CCB Interface

Figure 8-1 shows the CCB interface for both CCB address and data tenures. This interface formats CCB address tenures for the ECM transaction queue. It also contains the queueing and buffering needed to manage outstanding CCB data tenures. The buffers receive e500 core-initiated write and I/O-initiated read data (that hit in the L2/SRAM module) from the e500 write (128-bit wide) and read (128-bit wide) data buses and route them through the global data mux to the global data bus. The buffers also receive e500 core-initiated read and I/O-initiated write data (that hit in the L2/SRAM module) from the global data bus and forward them onto the CCB data bus (64 bits).

8.4 Initialization/Application Information

If 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 initialization. However, in any such system, one step near the end of the initialization routine must set EEBPCR[CPU_EN] to re-enable the e500 core. Note that for basic functionality, EEBPCR[CPU_EN] is the only field that must be written (provided a device other than the e500 core is used to initialize the device) in the ECM.

EEBPCR[CPU_PRI] specifies the priority level associated with all e500 core initiated transactions. This value allows users running time-critical applications to adjust the average response latency of transactions initiated by the core compared to those initiated by I/O masters. 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. Only transactions with a priority greater than the current CCB transaction can interrupt streaming. The higher the core’s priority, the lower the average latency needed for it to obtain bus grants from the ECM, because it can interrupt lower priority streaming. The default value of zero gives all core-initiated transactions the lowest priority, which prevents the core from interrupting I/O master transaction streams.

EEBACR[A_STRM_CNT] allows users to balance response latency with throughput and should prove useful in tuning systems with multiple time-critical tasks. The default value of 0b11 causes the ECM to attempt to stream as many as four transactions initiated from the same CCB master. Increasing this value increases the maximum number of transactions that may be streamed together from any one CCB master. Raising this value can increase throughput for high priority transactions, but may increase latency for lower priority transactions from another CCB master. Note that the e500 core must also have streaming enabled (through HID1[ASTME]) for the CCB to stream.

Chapter 9
DDR Memory Controller

9.1 Introduction

The fully programmable DDR SDRAM controller supports most first-generation JEDEC standard ×8 or ×16 DDR memories available, including buffered and unbuffered DIMMs. However, mixing unbuffered and registered DIMMs in the same system is not supported. Built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. A large set of special features, including DLL software override, crawl mode, and ECC error injection support rapid system debug.

NOTE

In this chapter, the word ‘bank’ refers to a physical bank specified by a chip select; ‘logical bank’ refers to one of the four sub-banks in each SDRAM chip. A sub-bank is specified by the two least significant bits (lsbs) of a bank address.

Figure 9-1 is a high-level block diagram of the DDR memory controller with its associated interfaces. Section 9.5, “Functional Description,” contains detailed figures of the controller.

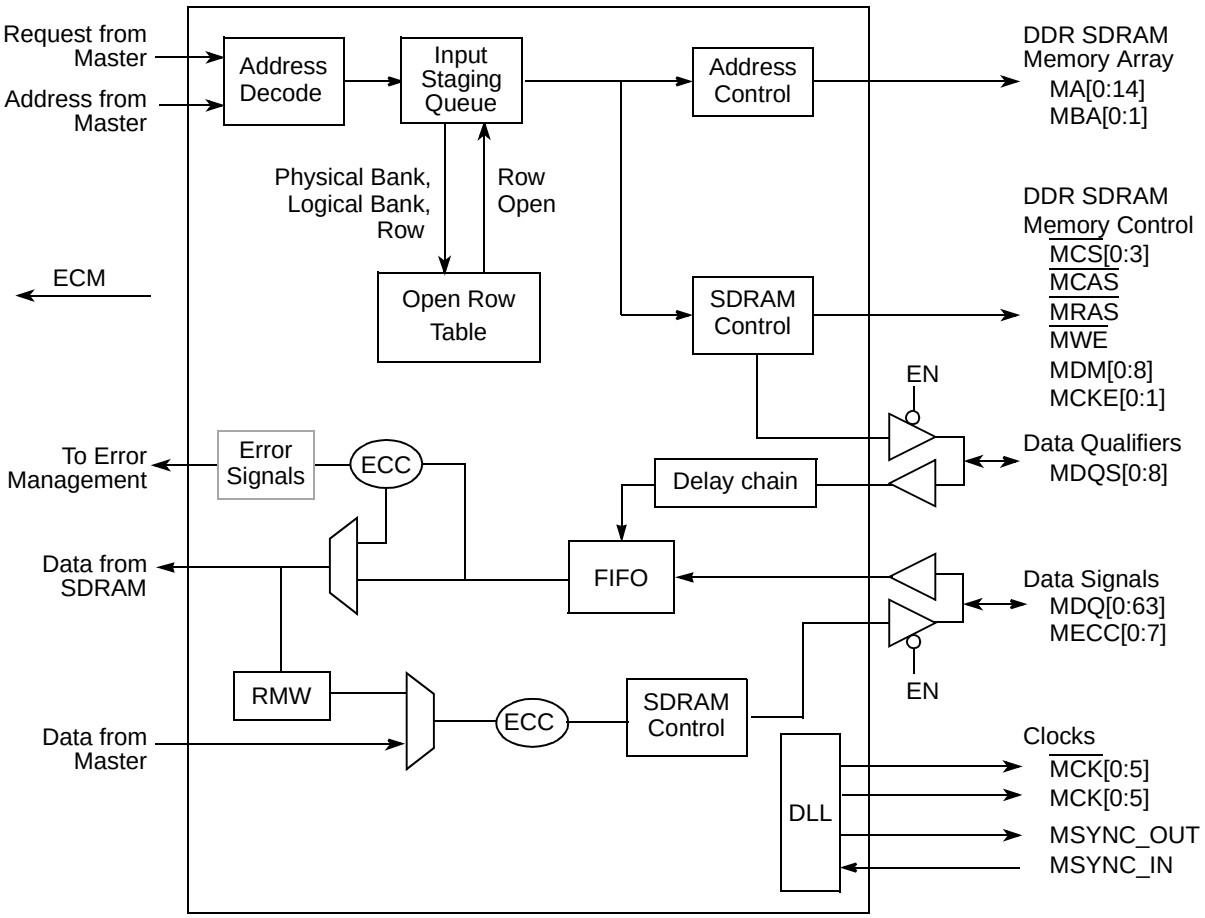


Figure 9-1. DDR Memory Controller Simplified Block Diagram

9.2 Features

The DDR memory controller includes these distinctive features:

- Support for DDR SDRAM
- 64-/72-bit SDRAM data bus
- Programmable settings for meeting all SDRAM timing parameters
- The following SDRAM configurations are supported
  - — As many as four physical banks (chip selects), each bank independently addressable
  - — 64-Mbit to 1-Gbit devices with ×8/×16 data ports (no direct ×4 support)
  - — Unbuffered and registered DIMMs
- Support for data mask signals and read-modify-write for sub-double word writes. Note that read-modify-write sequence is only necessary when ECC is enabled.
- Support for double-bit error detection and single-bit error correction ECC (8-bit check word across 64-bit data)

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

9.2.1 Modes of Operation

The DDR memory controller supports the following modes:

- Dynamic power management mode. The DDR memory controller can reduce power consumption by negating the SDRAM CKE signal when no transactions are pending to the SDRAM.
- Auto-precharge mode. Clearing DDR_SDRAM_INTERVAL[BSTOPRE] causes the memory controller to issue an auto precharge command with every read or write transaction. Auto precharge mode can be enabled for separate chip selects by setting CS*n*_CONFIG[AP_*n*_EN].

9.3 External Signal Descriptions

This section provides descriptions of the DDR memory controller's external signals. It describes each signal's behavior when the signal is asserted or negated and when the signal is an input or an output.

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.

9.3.1 Signals Overview

Memory controller signals are grouped as follows:

- Memory interface signals
- Clock signals
- Debug signals

Table 9-1 shows how DDR memory controller external signals are grouped. The *MPC8560 Integrated Communications Processor Hardware Specifications* has a pinout diagram showing pin numbers. It also lists all electrical and mechanical specifications.

Table 9-1. DDR Memory Interface Signal Summary

Name	Function/Description	Reset	Pins	I/O
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	One	1	O
MA[0:14]	Address bus	All zeros	15	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	One	1	O
$\overline{\text{MRAS}}$	Row address strobe	One	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]	DRAM clock enable	All zeros	2	O
MSYNC_IN	DRAM DLL synchronization in	Zero	1	I
MSYNC_OUT	DRAM DLL synchronization out	One	1	O
MDVAL	Memory debug data valid	Zero	1	O
MSRCID[0:4]	Memory debug source ID	All zeros	5	O

Table 9-2 shows the memory address signal mappings.

Table 9-2. Memory Address Signal Mappings

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

9.3.2 Detailed Signal Descriptions

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

9.3.2.1 Memory Interface Signals

Table 9-3 describes the DDR controller memory interface signals.

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

Signal(s)	I/O	Description	
MDQ[0:63]	I/O	Data bus. Both input and output signals on the DDR memory controller.	
	O	As outputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated—Represent the value of data being driven by the DDR memory controller.
		Timing	Assertion/Negation—Driven coincident with corresponding data strobes (MDQS) signal. High-impedance—No READ or WRITE command is in progress; data is not being driven by the memory controller or the DRAM.
	I	As inputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated—Represents the state of data being driven by the external DDR SDRAMs.
		Timing	Assertion/Negation—The DDR DIMM drives data during a READ transaction. High-impedance—No READ or WRITE command in progress; data is not being driven by the memory controller or the DRAM.
MDQS[0:8]	I/O	Data strobes. Inputs with read data and as outputs with write data.	
	O	As outputs, the data strobes are driven by the DDR memory controller during a write transaction. The memory controller always drives these signals low unless a read has been issued and incoming data strobes are expected. This keeps the data strobes from floating high when there are no transactions on the DRAM interface.	
		State Meaning	Asserted/Negated—Driven high when positive capture data is transmitted/received and driven low when negative capture data is transmitted/received. Centered in the data “eye” for writes; coincident with the data eye for reads. Treated as a clock. Data is valid when signals toggle. See Table 9-25 for byte lane assignments.
		Timing	Assertion/Negation—If a WRITE command is registered at clock edge *n*, data strobes at the DRAM assert centered in the data “eye” on clock edge *n* + 1. See the JEDEC DDR SDRAM specification for more information.
	I	As inputs, the data strobes are driven by the external DDR SDRAMs during a read transaction. The data strobes are used by the memory controller to synchronize data latching.	
		State Meaning	Asserted/Negated—Driven high when positive capture data is transmitted/received and driven low when negative capture data is transmitted/received. Centered in the data “eye” for writes; coincident with the data eye for reads. Treated as a clock. Data is valid when signals toggle. See Table 9-25 for byte lane assignments.
		Timing	Assertion/Negation—If a READ command is registered at clock edge *n*, and the latency is programmed in TIMING_CFG_1[CASLAT] to be *m* clocks, data strobes at the DRAM assert coincident with the data on clock edge *n* + *m*. See the JEDEC DDR SDRAM specification for more information.

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

Signal(s)	I/O	Description
MECC[0:7]	I/O	Error checking and correcting codes. Input and output signals for the DDR controller's bidirectional ECC bus. MECC[0:5] function in both normal and debug modes.
	O	As normal mode outputs the ECC signals represent the state of ECC driven by the DDR controller on writes. As debug mode outputs MECC[0:5] provide source ID and data-valid information. See Section 19.4.3.2, "Debug Information on ECC Pins," for more details.
		State Meaning Asserted/Negated—Represents the state of ECC being driven by the DDR controller on writes.
		Timing Assertion/Negation—Same timing as MDQ High-impedance—Same timing as MDQ
	I	As inputs, the ECC signals represent the state of ECC driven by the SDRAM devices on reads.
		State Meaning Asserted/Negated—Represents the state of ECC being driven by the DDR SDRAMs on reads.
		Timing Assertion/Negation—Same timing as MDQ High-impedance—Same timing as MDQ
MA[0:14]	O	Address bus. Memory controller outputs for the address to the DRAM. MA[0:14] carry 15 of the address bits for the DDR memory interface corresponding to the row and column address bits. MA[0] is the lsb of the address output from the memory controller.
		State Meaning Asserted/Negated—Represents the address driven by the DDR memory controller. Contains different portions of the address depending on the memory size and the DRAM command being issued by the memory controller. SeeTable 9-27 for a complete description of the mapping of these signals.
		Timing Assertion/Negation—The address is always driven when the memory controller is active. It is valid when a transaction is driven to DRAM (when $\overline{\text{MCS}}n$ is negated). High-impedance—When the memory controller is idle.
MBA[0:1]	O	Logical bank address. 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. Table 9-27 describes the mapping of these signals in all cases.
		Timing Assertion/Negation—Same timing as MA*n* High-impedance—Same timing as MA*n*
$\overline{\text{MCAS}}$	O	Column address strobe. Active-low SDRAM address multiplexing signal. $\overline{\text{MCAS}}$ is asserted for read or write transactions and for mode register set, refresh, and precharge commands.
		State Meaning Asserted—Indicates that a valid SDRAM column address is on the address bus for read and write transactions. See Table 9-28 for more information on the states required on $\overline{\text{MCAS}}$ for various other SDRAM commands. Negated—The column address is not guaranteed to be valid.
		Timing Assertion/Negation—Assertion and negation timing is directed by the values described in Section 9.4.1.3, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)." High-impedance—$\overline{\text{MCAS}}$ is always driven unless the memory controller is idle.

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

Signal(s)	I/O	Description	
$\overline{\text{MRAS}}$	O	Row address strobe. Active-low SDRAM address multiplexing signal. Asserted for activate commands. In addition; used for mode register set commands and refresh commands.	
		State Meaning	Asserted—Indicates that a valid SDRAM row address is on the address bus for read and write transactions. See Table 9-28 for more information on the states required on $\overline{\text{MRAS}}$ for various other SDRAM commands. Negated—The row address is not guaranteed to be valid.
		Timing	Assertion/Negation—Timing is directed by the values described in Section 9.4.1.3, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)." High impedance—$\overline{\text{MRAS}}$ is always driven unless the memory controller is idle.
$\overline{\text{MCS}}$[0:3]	O	Chip select. Four chip selects supported by the memory controller.	
		State Meaning	Asserted—Selects a physical SDRAM bank to perform a memory operation as described in Section 9.4.1.1, "Chip Select Memory Bounds (CSn_BNDS)," and Section 9.4.1.2, "Chip Select Configuration (CSn_CONFIG)." The DDR controller asserts one of the $\overline{\text{MCS}}$[0:3] signals to begin a memory cycle. Negated—Indicates no SDRAM action during the current cycle
		Timing	Assertion/Negation—Asserted to signal any new transaction to the SDRAM. The transaction must adhere to the timing constraints set in TIMING_CFG_1. High impedance—Always driven unless the memory controller is disabled
$\overline{\text{MWE}}$	O	Write enable. Asserted when a write transaction is issued to the SDRAM. This is also used for mode registers set commands and precharge commands.	
		State Meaning	Asserted—Indicates a memory write operation. See Table 9-28 for more information on the states required on $\overline{\text{MWE}}$ for various other SDRAM commands. Negated—Indicates a memory read operation
		Timing	Assertion/Negation—Similar timing as $\overline{\text{MRAS}}$ and $\overline{\text{MCAS}}$. Used for write commands. High impedance—$\overline{\text{MWE}}$ is always driven unless the memory controller is idle.
MDM[0:8]	O	DDR SDRAM data output mask. Masks 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 byte (MSB); MDM7 corresponds to the LSB. MDM8 corresponds to the ECC byte. Table 9-25 shows byte lane encodings.	
		State Meaning	Asserted—Prevents writing to DDR SDRAM. Asserted when data is written to DRAM if the corresponding byte(s) should be masked for the write. Note that the MDM*n* signals are active-high for the DDR controller. MDM*n* is part of the DDR command encoding. Negated—Allows the corresponding byte to be read from or written to the SDRAM
		Timing	Assertion/Negation—Same timing as MDQx as outputs High impedance—Always driven unless the memory controller is disabled

9.3.2.2 Clock Interface Signals

Table 9-4 contains the detailed descriptions of the clock signals of the DDR controller.

Table 9-4. Clock Signals—Detailed Signal Descriptions

Signal(s)	I/O	Description	
MCK[0:5], $\overline{\text{MCK}}$[0:5]	O	DRAM clock outputs and their complements. See Section 9.5.4.1, “Clock Distribution.”	
		State Meaning	Asserted/Negated—The JEDEC DDR SDRAM specifications require true and complement clocks. A clock edge is seen by the SDRAM when the true and complement cross.
		Timing	Assertion/Negation—Should be synchronized at the SDRAM with the internal clocks of the MPC8560. This is done with the DLL.
MCKE[0:1]	O	Clock enable. Two identical output signals (each hereafter referred to simply as MCKE) used as the clock enable to one or more SDRAMs. 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 signal transitions on MCK or $\overline{\text{MCK}}$. MCK/$\overline{\text{MCK}}$ are don’t cares while MCKE is negated.
		Timing	Assertion/Negation—Similar timing to MA*n* High impedance—Always driven
MSYNC_IN	I	DRAM DLL synchronization input. The DLL uses this input from the feedback loop to synchronize SDRAM clocks with the internal clocks of the MPC8560. See Section 9.5.4.1, “Clock Distribution.”	
		State Meaning	Asserted/Negated—Toggles at the same frequency as the applied SDRAM clocks (MCKx), but is not in phase with MCK*n*
		Timing	Assertion/Negation—This is used for locking the DLL.
MSYNC_OUT	O	DRAM DLL synchronization output. Output for the DLL feedback loop. See Section 9.5.4.1, “Clock Distribution.”	
		State Meaning	Asserted/Negated—See MSYNC_IN description

9.3.2.3 Debug Signals

The debug signals MSRCID[0:4] and MDVAL have no function in normal DDR controller operation. A detailed description of these signals can be found in Section 19.4.3, “DDR SDRAM Interface Debug.”

9.4 Memory Map/Register Definition

Table 9-5 shows the register memory map for the DDR memory controller. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 9-5. DDR Memory Controller Memory Map

Offset	Register	Access	Reset	Section/Page
0x0_2000	CS0_BNDS—Chip select 0 memory bounds	R/W	0x0000_0000	9.4.1.1/9-10
0x0_2008	CS1_BNDS—Chip select 1 memory bounds			
0x0_2010	CS2_BNDS—Chip select 2 memory bounds			
0x0_2018	CS3_BNDS—Chip select 3 memory bounds			
0x0_2080	CS0_CONFIG—Chip select 0 configuration	R/W	0x0000_0000	9.4.1.2/9-10
0x0_2084	CS1_CONFIG—Chip select 1 configuration			
0x0_2088	CS2_CONFIG—Chip select 2 configuration			
0x0_208C	CS3_CONFIG—Chip select 3 configuration			
0x0_2108	TIMING_CFG_1—DDR SDRAM timing configuration 1	R/W	0x0000_0000	9.4.1.3/9-11
0x0_210C	TIMING_CFG_2—DDR SDRAM timing configuration 2	R/W	0x0000_0000	9.4.1.4/9-13
0x0_2110	DDR_SDRAM_CFG—DDR SDRAM control configuration	R/W	0x0200_0000	9.4.1.5/9-14
0x0_2118	DDR_SDRAM_MODE—DDR SDRAM mode configuration	R/W	0x0000_0000	9.4.1.6/9-16
0x0_2124	DDR_SDRAM_INTERVAL—DDR SDRAM interval configuration	R/W	0x0000_0000	9.4.1.7/9-16
0x0_2E00	DATA_ERR_INJECT_HI—Memory data path error injection mask high	R/W	0x0000_0000	9.4.1.8/9-17
0x0_2E04	DATA_ERR_INJECT_LO—Memory data path error injection mask low	R/W	0x0000_0000	9.4.1.9/9-18
0x0_2E08	ECC_ERR_INJECT—Memory data path error injection mask ECC	R/W	0x0000_0000	9.4.1.10/9-18
0x0_2E20	CAPTURE_DATA_HI—Memory data path read capture high	R/W	0x0000_0000	9.4.1.11/9-19
0x0_2E24	CAPTURE_DATA_LO—Memory data path read capture low	R/W	0x0000_0000	9.4.1.12/9-20
0x0_2E28	CAPTURE_ECC—Memory data path read capture ECC	R/W	0x0000_0000	9.4.1.13/9-20
0x0_2E40	ERR_DETECT—Memory error detect	Special	0x0000_0000	9.4.1.14/9-21
0x0_2E44	ERR_DISABLE—Memory error disable	R/W	0x0000_0000	9.4.1.15/9-21
0x0_2E48	ERR_INT_EN—Memory error interrupt enable	R/W	0x0000_0000	9.4.1.16/9-22
0x0_2E4C	CAPTURE_ATTRIBUTES—Memory error attributes capture	R/W	0x0000_0000	9.4.1.17/9-23
0x0_2E50	CAPTURE_ADDRESS—Memory error address capture	R/W	0x0000_0000	9.4.1.18/9-24
0x0_2E58	ERR_SBE—Single-Bit ECC memory error management	R/W	0x0000_0000	9.4.1.19/9-25

9.4.1 Register Descriptions

This section describes the DDR memory controller registers. Shading indicates reserved fields that should not be written.

9.4.1.1 Chip Select Memory Bounds (CS*n*_BNDS)

The chip select bounds registers (CS*n*_BNDS) shown in Figure 9-2 define the starting and ending address of the memory space that corresponds to the individual chip selects. Note that the size specified in CS*n*_BNDS should equal the size of physical DRAM. Also, note that EA*n* must be greater than or equal to SA*n*. If the high-order 8 bits of an address are greater than or equal to SA*n*, and they are less than or equal to EA*n*, then chip select *n* will be used.

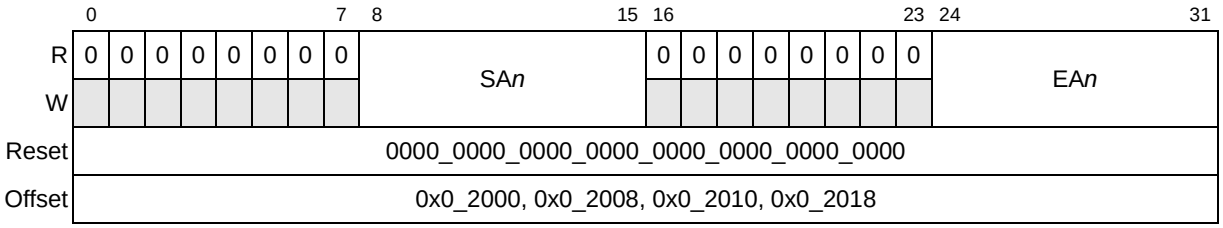


Figure 9-2. Chip Select Bounds Registers (CS*n*_BNDS)

Table 9-6 describes the CS*n*_BNDS register fields.

Table 9-6. CS*n*_BNDS Field Descriptions

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

9.4.1.2 Chip Select Configuration (CS*n*_CONFIG)

The chip select configuration (CS*n*_CONFIG) registers shown in Figure 9-3 enable the DDR chip selects and set the number of row and column bits used for each chip select. These registers should be loaded with the correct number of row and column bits for each SDRAM. Because CS*n*_CONFIG[ROW_BITS_CS_*n*,COL_BITS_CS_*n*] establish address multiplexing, the user should take great care to set these values correctly.

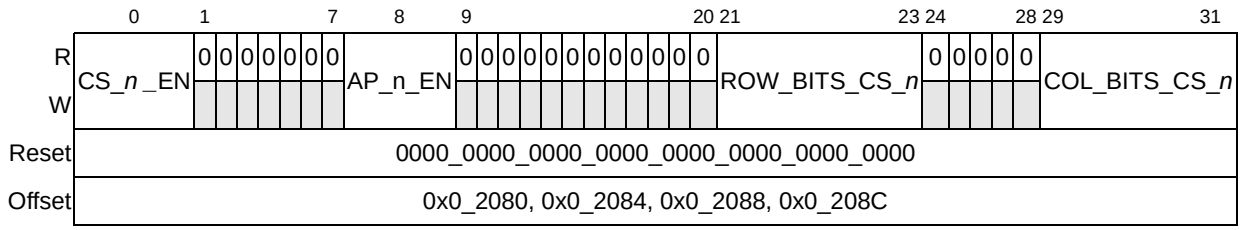


Figure 9-3. Chip Select Configuration Register (CS*n*_CONFIG)

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

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

Bits	Name	Description
0	CS_*n*_EN	Chip select *n* enable 0 Chip select *n* is not active. 1 Chip select *n* is active and assumes the state set in CSn_BNDS.
1–7	—	Reserved
8	AP_n_EN	Chip select *n* auto precharge enable 0 Chip select n is auto precharged only if global auto precharge mode is enabled (DDR_SDRAM_INTERVAL[BSTOPRE] = 0). 1 Chip select *n* always issues an auto precharge for read and write transactions.
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 011–111 Reserved

9.4.1.3 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.

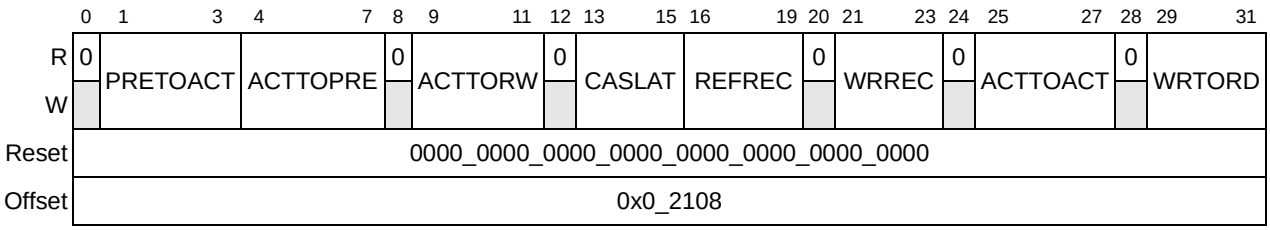


Figure 9-4. DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1)

Table 9-8 describes TIMING_CFG_1 fields.

Table 9-8. TIMING_CFG_1 Field Descriptions

Bits	Name	Description
0	—	Reserved, should be cleared.
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 100 4 clocks 001 1 clock 101 5 clocks 010 2 clocks 110 6 clocks 011 3 clocks 111 7 clocks
4–7	ACTTOPRE	Activate to precharge interval (t_{ras}). Determines the number of clock cycles from an activate command until a precharge command is allowed. 0000 Reserved 0010 2 clocks 0001 1 clock ... 0010 2 clocks 1111 15 clocks
8	—	Reserved, should be cleared.
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 100 4 clocks 001 1 clock 101 5 clocks 010 2 clocks 110 6 clocks 011 3 clocks 111 7 clocks
12	—	Reserved, should be cleared.
13–15	CASLAT	$\overline{\text{MCAS}}$ latency from READ command. Number of clock cycles between registration of a READ command by the SDRAM and the availability of the first output data. If a READ command is registered at clock edge n and the latency is m clocks, data is available nominally coincident with clock edge $n + m$. This value must be programmed at initialization as described in Section 9.4.1.6, "DDR SDRAM Mode Configuration (DDR_SDRAM_MODE)." 000 Reserved 100 2.5 clocks 001 1 clock 101 3 clocks 010 1.5 clocks 110 3.5 clocks 011 2 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 REFREC value. 0000 Reserved 0011 11 clocks 0001 9 clocks ... 0010 10 clocks 1111 23 clocks
20	—	Reserved, should be cleared.
21–23	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. 000 0 clock 100 4 clocks 001 1 clock 101 5 clocks 010 2 clocks 110 6 clocks 011 3 clocks 111 7 clocks
24	—	Reserved, should be cleared.

Table 9-8. TIMING_CFG_1 Field Descriptions (continued)

Bits	Name	Description
25–27	ACTTOACT	Activate-to-activate interval (t_{rrd}). Number of clock cycles from an activate command until another activate command is allowed for a different logical bank in the same physical bank (chip select). 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101–111 Reserved
28	—	Reserved, should be cleared.
29–31	WRTORD	Last write data pair to read command issue interval (t_{wtr}). Number of clock cycles between the last write data pair and the subsequent read command to the same physical bank. 000 Reserved 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks

9.4.1.4 DDR SDRAM Timing Configuration 2 (TIMING_CFG_2)

DDR SDRAM timing configuration 2, shown in Figure 9-5, sets the clock delay to data for writes.

	0			3	4			7	8			11	12	13					18	19			21	22								31
R	0	0	0	0	CPO				0	0	0	0	ACSM	0	0	0	0	0	0	WR_DATA_DELAY				0	0	0	0	0	0	0	0	0	0
W																																
Reset	0000_0000_0000_0000_0000_0000_0000_0000																															
Offset	0x0_210C																															

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

Table 9-9 describes the TIMING_CFG_2 fields.

Table 9-9. TIMING_CFG_2 Register Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–7	CPO	$\overline{\text{MCAS}}$-to-preamble override. Defines the number of DRAM cycles between when a read is issued and when the corresponding DQS preamble is valid for the memory controller. 0000 Default. $\overline{\text{MCAS}}$ to preamble is defined as CASLAT + 1 0001 ⌈CASLAT⌉ 0010 ⌈CASLAT⌉ + 1/2 0011 ⌈CASLAT⌉ + 1 0100 ⌈CASLAT⌉ + 3/2 0101 ⌈CASLAT⌉ + 2 0110 ⌈CASLAT⌉ + 5/2 0111 ⌈CASLAT⌉ + 3 1000 ⌈CASLAT⌉ + 7/2 1001 ⌈CASLAT⌉ + 4 1010 ⌈CASLAT⌉ + 9/2 1011 ⌈CASLAT⌉ + 5 1100–1111 Reserved
8–11	—	Reserved

Table 9-9. TIMING_CFG_2 Register Field Descriptions (continued)

Bits	Name	Description
12	ACSM	Address and control shift mode 0 The DRAM address and control buses are output in the default mode. 1 The DRAM address and control buses are delayed by 1/2 DRAM cycle before being driven onto the pins.
13–18	—	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 101–111 Reserved
22–31	—	Reserved

9.4.1.5 DDR SDRAM Control Configuration (DDR_SDRAM_CFG)

The DDR SDRAM control configuration register, shown in Figure 9-6, enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting, registered DIMMS, and dynamic power management.

	0	1	2	3	4	5	6	7	8	9	10	11	13	14	15
R	MEM_EN	SREN	ECC_EN	RD_EN	0	0	SDRAM_TYPE		0	0	DYN_PWR	0	0	0	NCAP	0
W																
Reset	0000_0010_0000_0000															

	16	17														31
	2T_EN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0000_0000_0000_0000															
Offset	0x0_2110															

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

Table 9-10 describes the DDR_SDRAM_CFG fields.

Table 9-10. DDR_SDRAM_CFG Field Descriptions

Bits	Name	Description
0	MEM_EN	DDR SDRAM interface logic enable 0 SDRAM interface logic is disabled 1 SDRAM interface logic is enabled. Must not be set until all other memory configuration parameters have been appropriately configured by initialization code.
1	SREN	Self refresh enable (during sleep) 0 SDRAM self refresh is disabled during sleep 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. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, ERR_DISABLE[MBED] must be zero and ERR_INT_EN[MBEE] and ECC_EN must be one to ensure an interrupt is generated. See Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).” 0 No ECC errors are reported. No ECC interrupts are generated. 1 ECC is enabled.
3	RD_EN	Registered DIMM enable. Specifies the type of DIMM used in the system. 0 Indicates unbuffered DIMMs. 1 Indicates registered DIMMs.
4–5	—	Reserved
6–7	SDRAM_TYPE	Type of SDRAM device to be used 00–01 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.
11–13	—	Reserved
14	NCAP	Non-concurrent auto precharge 0 SDRAMs in system support concurrent auto precharge. 1 SDRAMs in system do not support concurrent auto precharge.
15	—	Reserved
16	2T_EN	2T timing enable 0 1T timing is used. The SDRAM command/address are held for only 1 cycle on the SDRAM bus. 1 2T timing is enabled. The SDRAM command/address are held for 2 full cycles on the SDRAM bus for every SDRAM transaction. However, the chip select is only held for the second cycle.
17–31	—	Reserved

9.4.1.6 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.

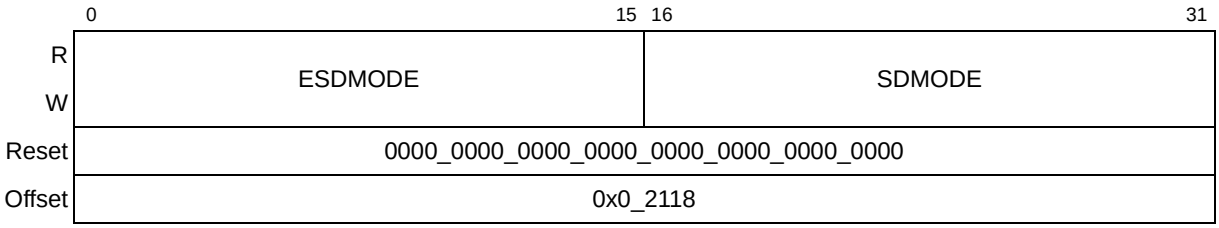


Figure 9-7. DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)

Table 9-11 describes the DDR_SDRAM_MODE fields.

Table 9-11. DDR_SDRAM_MODE Field Descriptions

Bits	Name	Description
0–15	ESDMODE [0:15]	Extended SDRAM mode. Specifies the initial value loaded into the DDR SDRAM extended mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. The value of ESDMODE[1:15] is driven onto MA[14:0] during the EXTENDED MODE REGISTER SET operation of the intialization sequence. The lsb of ESDMODE (ESDMODE[15]) is driven onto MA[0] and ESDMODE[1] is driven onto MA[14].
16–31	SDMODE [0:15]	SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. The value of SDMODE[1:15] is driven onto MA[14:0] during the MODE REGISTER SET operation of the intialization sequence. The lsb of SDMODE (SDMODE[15]) is driven onto MA[0] and SDMODE[1] is driven onto MA[14]. Because the memory controller forces SDMODE[3–8] to certain values depending upon the state of the initialization sequence (for resetting the SDRAM's DLL) the corresponding bits of this field is ignored by the memory controller.

9.4.1.7 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.

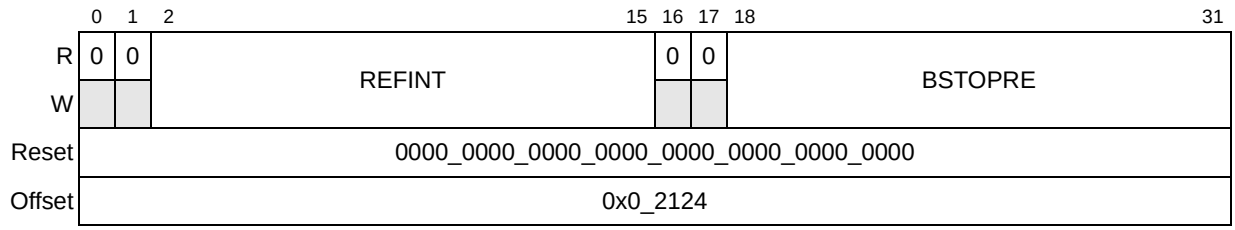


Figure 9-8. DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)

Table 9-12 describes the DDR_SDRAM_INTERVAL fields.

Table 9-12. 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. Note that REFINT must be set to a non-zero value in order for the DDR to enter sleep mode. See Section 17.5.1.5.3, “Sleep Mode,” for additional details.
16–17	—	Reserved
18–31	BSTOPRE	Precharge interval. Sets the duration (in memory bus clocks) that a page is retained after a DDR SDRAM access. If BSTOPRE is zero, the DDR memory controller uses auto precharge read and write commands rather than operating in page mode. This is called global auto precharge mode.

9.4.1.8 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.

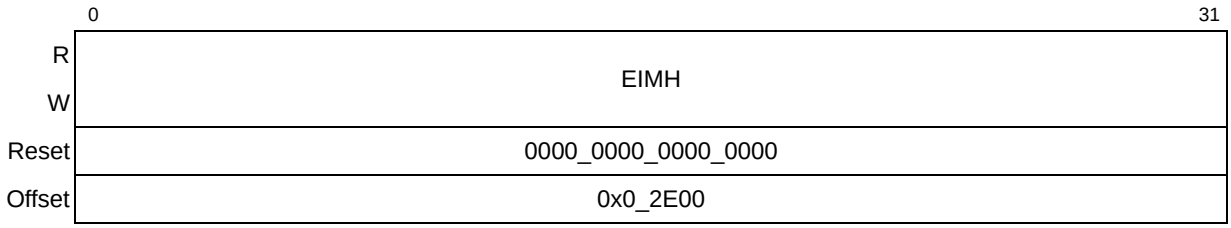


Figure 9-9. Memory Data Path Error Injection Mask High Register (DATA_ERR_INJECT_HI)

Table 9-13 describes the DATA_ERR_INJECT_HI fields.

Table 9-13. DATA_ERR_INJECT_HI Field Descriptions

Bits	Name	Description
0–31	EIMH	Error injection mask high data path. Used to test ECC by forcing errors on the high word of the data path. Setting a bit causes the corresponding data path bit to be inverted on memory bus writes.

9.4.1.9 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.

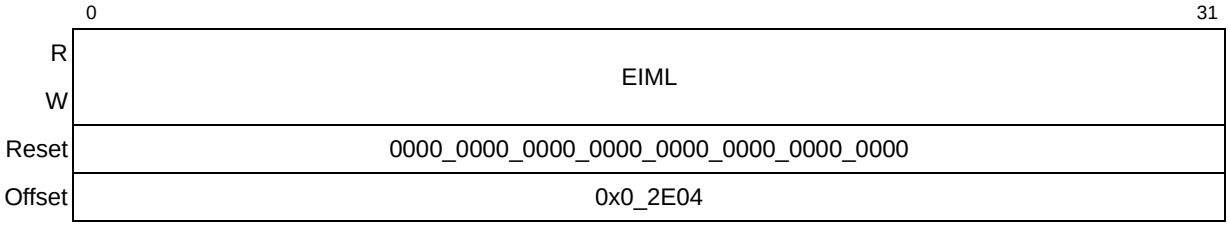


Figure 9-10. Memory Data Path Error Injection Mask Low Register (DATA_ERR_INJECT_LO)

Table 9-14 describes the DATA_ERR_INJECT_LO fields.

Table 9-14. DATA_ERR_INJECT_LO Field Descriptions

Bits	Name	Description
0–31	EIML	Error injection mask low data path. Used to test ECC by forcing errors on the low word of the data path. Setting a bit causes the corresponding data path bit to be inverted on memory bus writes.

9.4.1.10 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.

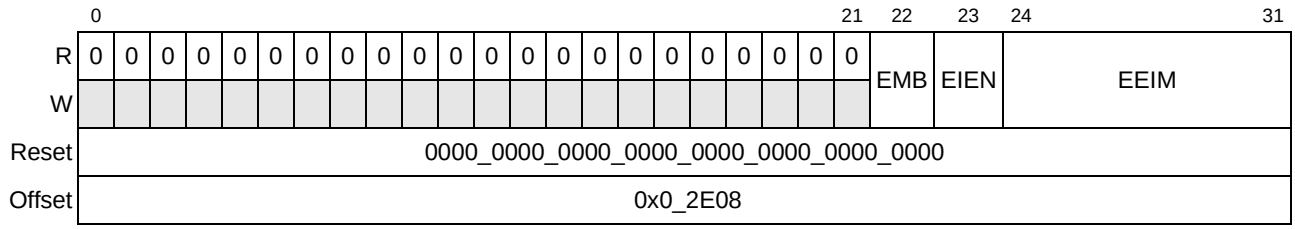


Figure 9-11. Memory Data Path Error Injection Mask ECC Register (ECC_ERR_INJECT)

Table 9-15 describes the ECC_ERR_INJECT fields.

Table 9-15. 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.

9.4.1.11 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.

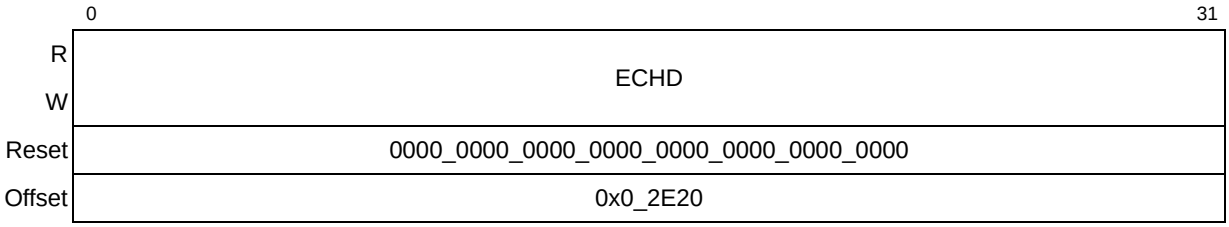


Figure 9-12. Memory Data Path Read Capture High Register (CAPTURE_DATA_HI)

Table 9-16 describes the CAPTURE_DATA_HI fields.

Table 9-16. CAPTURE_DATA_HI Field Descriptions

Bits	Name	Description
0–31	ECHD	Error capture high data path. Captures the high word of the data path when errors are detected.

9.4.1.12 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.

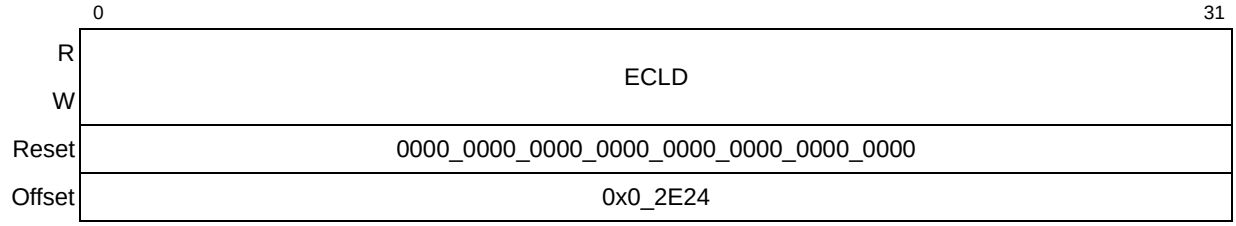


Figure 9-13. Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO)

Table 9-17 describes the CAPTURE_DATA_LO fields.

Table 9-17. CAPTURE_DATA_LO Field Descriptions

Bits	Name	Description
0–31	ECLD	Error capture low data path. Captures the low word of the data path when errors are detected.

9.4.1.13 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.

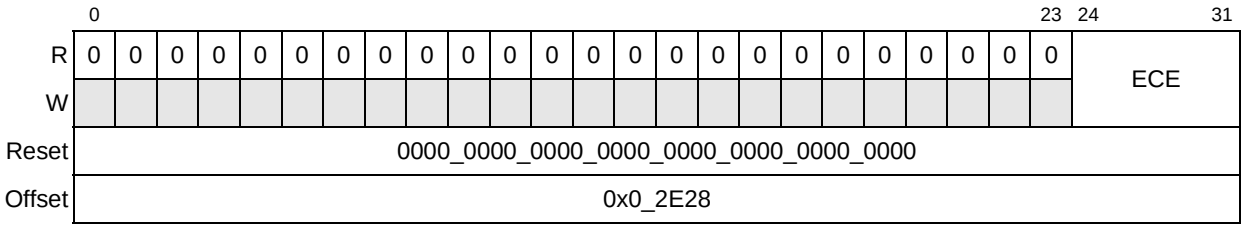


Figure 9-14. Memory Data Path Read Capture ECC Register (CAPTURE_ECC)

Table 9-18 describes the CAPTURE_ECC fields.

Table 9-18. 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.

9.4.1.14 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 is disabled with ERR_DISABLE, the corresponding error is never detected or captured in ERR_DETECT.

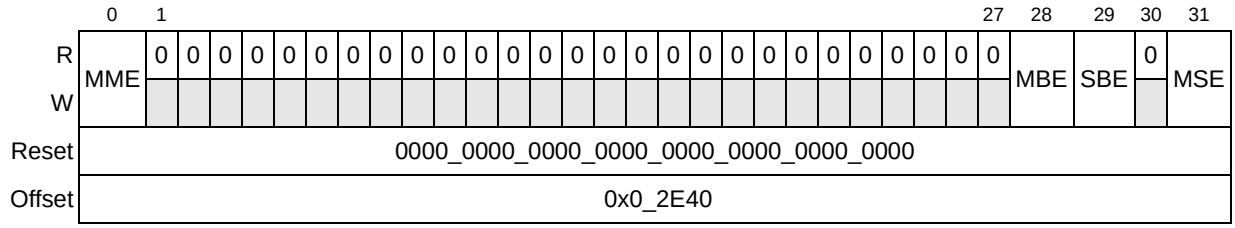


Figure 9-15. Memory Error Detect Register (ERR_DETECT)

Table 9-19 describes the ERR_DETECT fields.

Table 9-19. 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 threshold set in ERR_SBE[SBET]. 1 The number of single-bit ECC errors detected crossed the threshold set in ERR_SBE[SBET].
30	—	Reserved
31	MSE	Memory select error. This bit is cleared by software writing a 1. 0 A memory select error has not been detected. 1 A memory select error has been detected.

9.4.1.15 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.

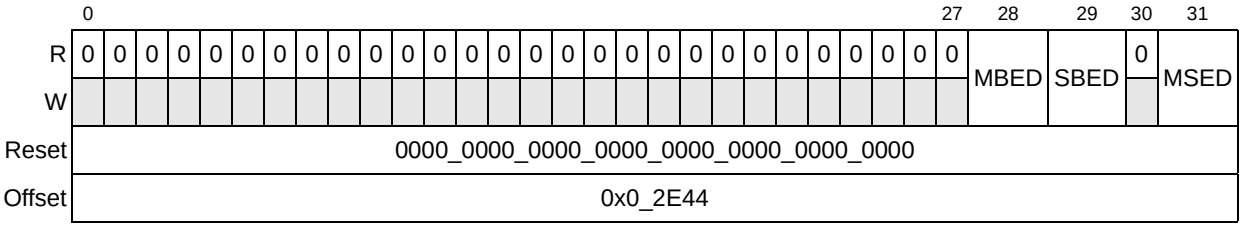


Figure 9-16. Memory Error Disable Register (ERR_DISABLE)

Table 9-20 describes the ERR_DISABLE fields.

Table 9-20. 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. They are reported if ERR_INT_EN[MBEE] is set. Note that uncorrectable read errors cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, MBED must be zero and ERR_INT_EN[MBEE] and ECC_EN must be one to ensure that an interrupt is generated. 1 Multiple-bit ECC errors are not detected or reported.
29	SBED	Single-bit ECC error disable 0 Single-bit ECC errors are enabled. 1 Single-bit ECC errors are disabled.
30	—	Reserved
31	MSED	Memory select error disable 0 Memory select errors are enabled. 1 Memory select errors are disabled.

9.4.1.16 Memory Error Interrupt Enable (ERR_INT_EN)

The memory error interrupt enable register, shown in Figure 9-17, enables ECC interrupts or memory select error interrupts. When an enabled interrupt condition occurs, the internal *int* signal is asserted to the programmable interrupt controller (PIC).

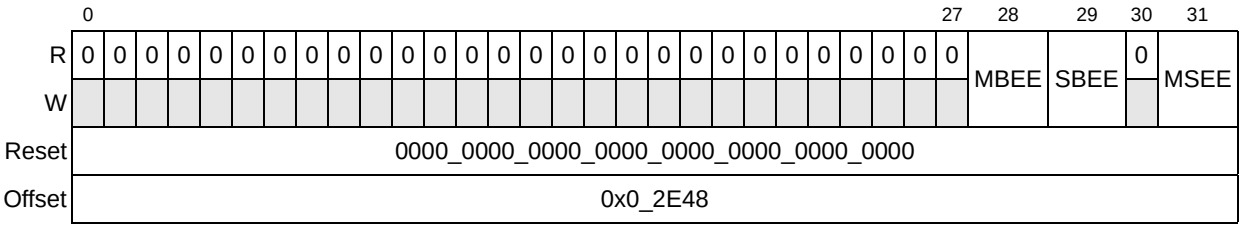


Figure 9-17. Memory Error Interrupt Enable Register (ERR_INT_EN)

Table 9-21 describes the ERR_INT_EN fields.

Table 9-21. ERR_INT_EN Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	MBEE	Multiple-bit ECC error interrupt enable. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, ERR_DISABLE[MBED] must be zero and MBEE and DDR_SDRAM_CFG[ECC_EN] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).” 0 Multiple-bit ECC errors cannot generate interrupts. 1 Multiple-bit ECC errors generate interrupts.
29	SBEE	Single-bit ECC error interrupt enable 0 Single-bit ECC errors cannot generate interrupts. 1 Single-bit ECC errors generate interrupts.
30	—	Reserved
31	MSEE	Memory select error interrupt enable 0 Memory select errors do not cause interrupts. 1 Memory select errors generate interrupts.

9.4.1.17 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.

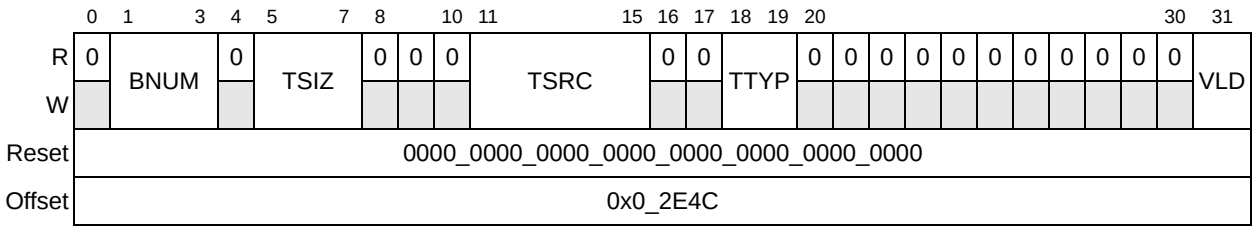


Figure 9-18. Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES)

Table 9-22 describes the CAPTURE_ATTRIBUTES fields.

Table 9-22. CAPTURE_ATTRIBUTES Field Descriptions

Bits	Name	Description
0	—	Reserved
1–3	BNUM	Data beat number. 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 9-22. CAPTURE_ATTRIBUTES Field Descriptions (continued)

Bits	Name	Description
11–15	TSRC	Transaction source for the error 00000 PCI 00001–00011 Reserved 00100 Local bus 00101–00111 Reserved 01000 Configuration space 01001 Reserved 01010 Boot sequencer 01011 Reserved 01100 RapidIO 01101–01111 Reserved 10000 Processor (instruction) 10001 Processor (data) 10010–10011 Reserved 10100 CPM 10101 DMA 10110 Reserved 10111 SAP 11000 TSEC1 11001 TSEC2 11010 Reserved 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	Valid. Set as soon as valid information is captured in the error capture registers.

9.4.1.18 Memory Error Address Capture (CAPTURE_ADDRESS)

The memory error address capture register, shown in Figure 9-19, holds the 32 lsbs of a transaction when a DDR ECC error is detected.

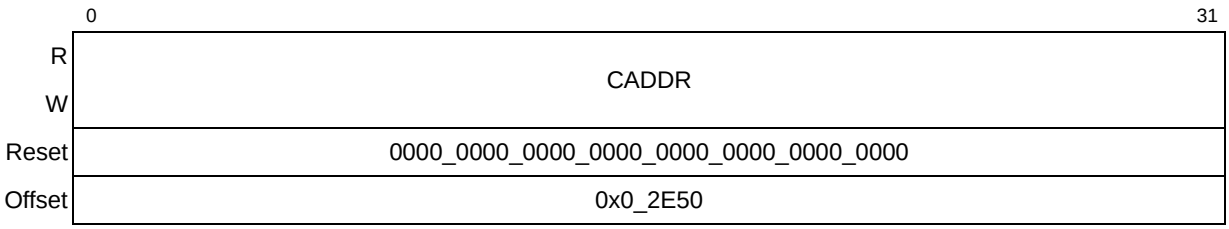


Figure 9-19. Memory Error Address Capture Register (CAPTURE_ADDRESS)

Table 9-23 describes the CAPTURE_ADDRESS fields.

Table 9-23. CAPTURE_ADDRESS Field Descriptions

Bits	Name	Description
0–31	CADDR	Captured address. Captures the 32 lsbs of the transaction address when an error is detected.

9.4.1.19 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. When the counter field reaches the threshold, it wraps back to the reset value (0). If necessary, software must clear the counter after it has managed the error.

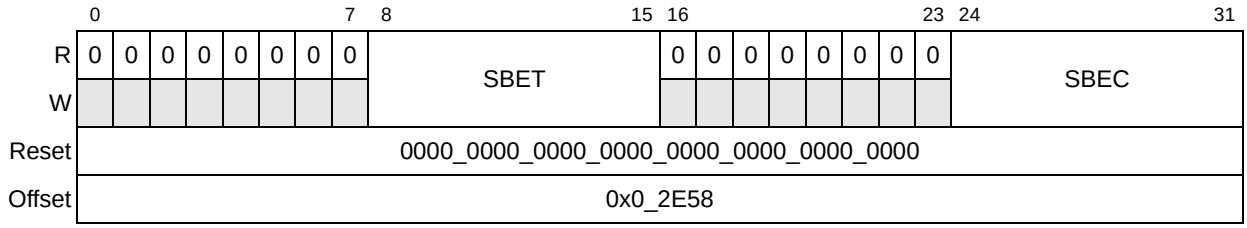


Figure 9-20. Single-Bit ECC Memory Error Management Register (ERR_SBE)

Table 9-24 describes the ERR_SBE fields.

Table 9-24. ERR_SBE Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	SBET	Single-bit error threshold. Establishes the number of single-bit errors that must be detected before an error condition is reported.
16–23	—	Reserved
24–31	SBEC	Single-bit error counter. Indicates the number of single-bit errors detected and corrected since the last error report. If single-bit error reporting is enabled, an error is reported and an interrupt is generated when this value equals SBET. SBEC is automatically cleared when the threshold value is reached.

9.5 Functional Description

The DDR SDRAM controller controls processor and I/O interactions with system memory. It provides support for JEDEC-compliant 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.

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 input queue entry.

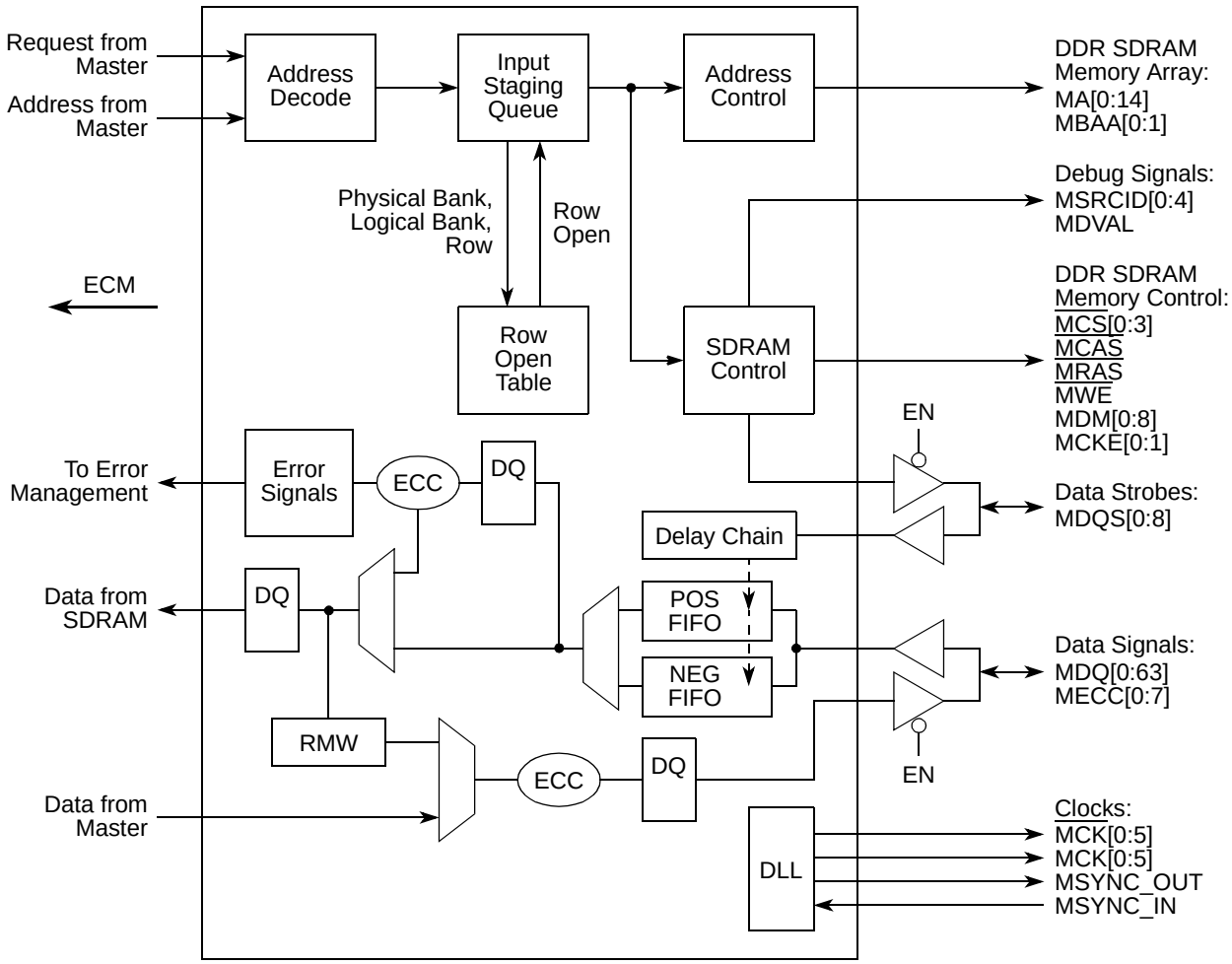


Figure 9-21. DDR Memory Controller Block Diagram

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].

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 command followed by a READ or WRITE. (Accessing open pages does not require an ACTIVE command.) The address bits registered coincident with the activate command specifies the logical bank and row to be accessed. The address coincident with the READ or WRITE command specify the logical bank and starting column for the burst access.

The data interface is source synchronous, meaning whatever sources the data also provides a clocking signal to synchronize data reception. These bidirectional data strobes (MDQS[0:8]) are inputs to the controller during reads, and outputs during writes. The DDR SDRAM specification requires the data strobe signals to be centered within the data tenure during writes and to be offset by the controller to the center of the data tenure during reads. This 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 how the on-chip DLL can be used with the memory controller.

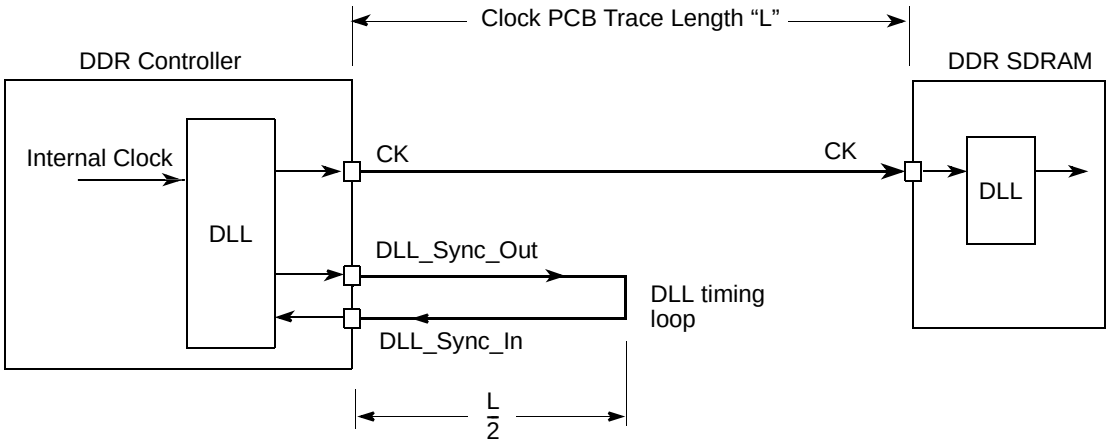


Figure 9-22. Controller DLL Timing Loop

Figure 9-23 shows an example DDR SDRAM configuration with four physical banks.

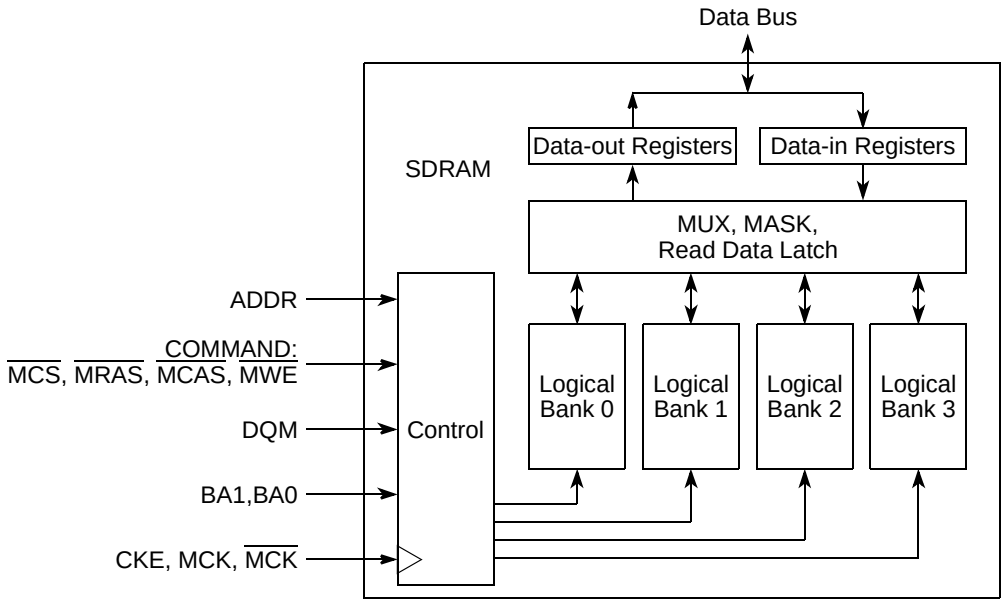


Figure 9-23. Typical Dual Data Rate SDRAM Internal Organization

Figure 9-24 shows some typical signal connections.

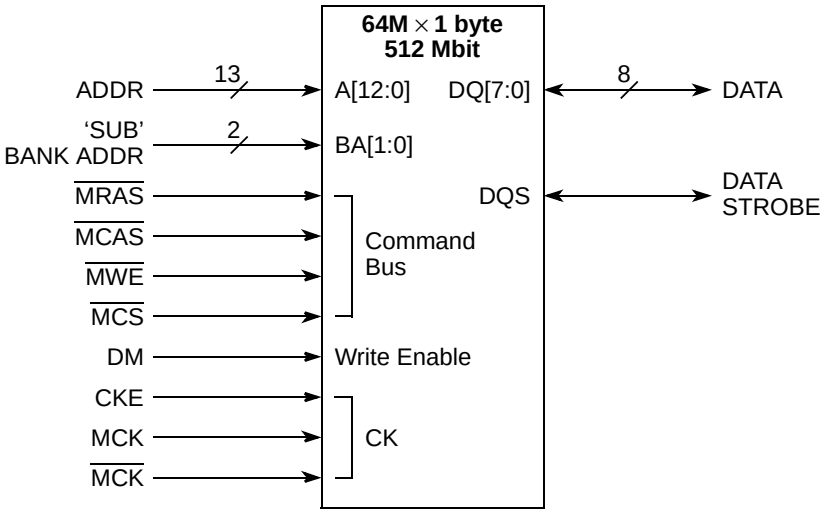
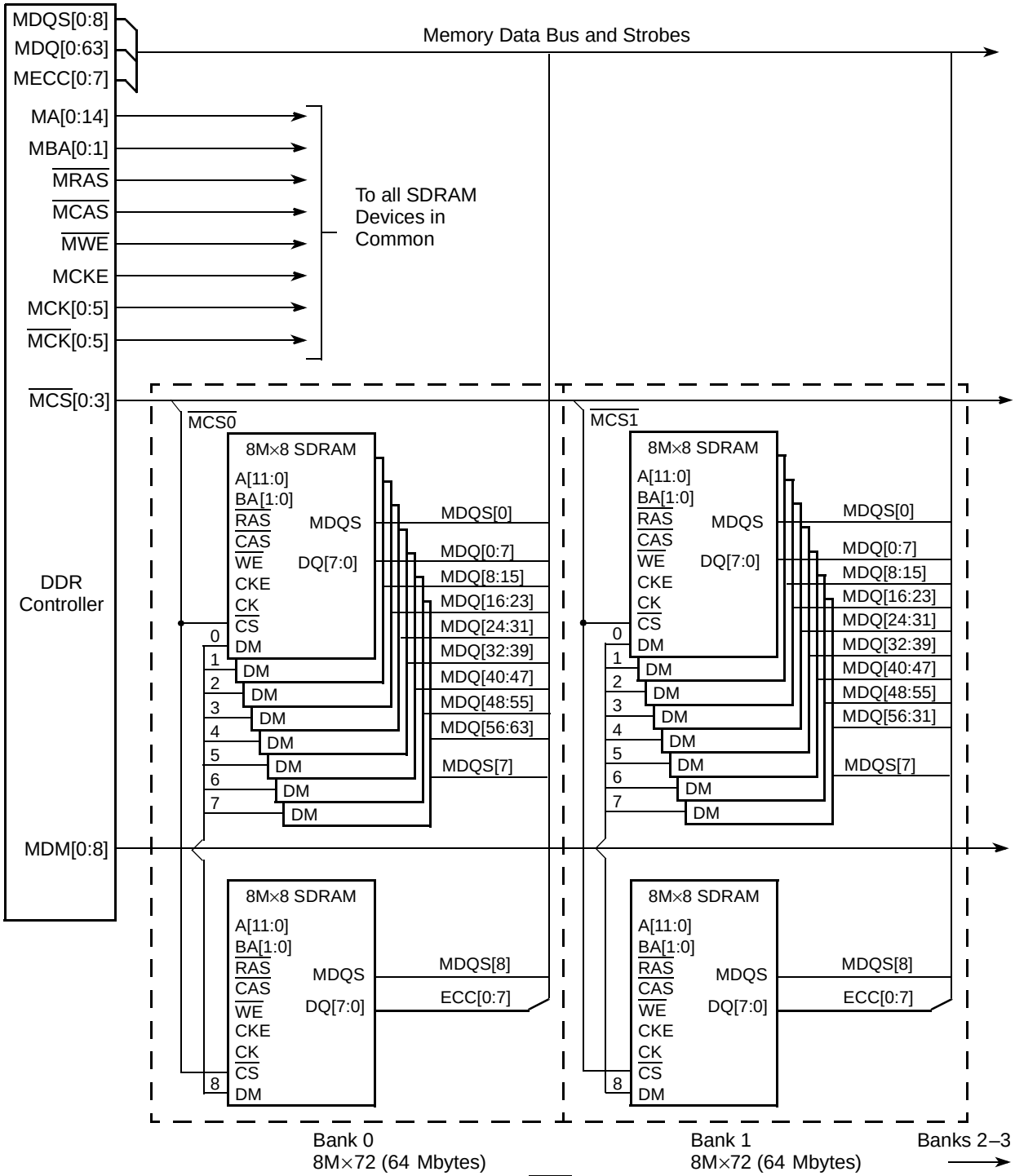


Figure 9-24. Typical DDR SDRAM Interface Signals

Figure 9-25 shows an example DDR SDRAM configuration with four physical banks each comprised of nine 8M × 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 15 address pins, but in this example the DDR SDRAM devices use only 12 bits.



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

Figure 9-25. Example 256-Mbyte DDR SDRAM Configuration with ECC

Section 9.5.12, “Error Management,” explains how the DDR memory controller handles errors.

9.5.1 DDR SDRAM Interface Operation

The DDR memory controller supports many different DDR SDRAM configurations. SDRAMs with different sizes can be used in the same system. Fourteen multiplexed address signals and two logical bank select signals support 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 bits 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, 128, 256, and 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 8 data signals (DQ[0:7]). A 16-bit DDR SDRAM device has 2 DQM signals associated with specific halves of the 16 data signals (DQ[0:7] and DQ[8:15]).

When ECC is enabled, all memory accesses are performed on double-word boundaries (that is, all DQM signals are set simultaneously). However, when ECC is disabled, the memory system uses the DQM signals for byte lane selection.

Table 9-25 shows the DDR memory controller’s relationships between data byte lane 0–7, MDM[0:7], MDQS[0:7], and MDQ[0:63].

Table 9-25. Byte Lane to Data Relationship

Data Byte Lane	Data Bus Mask	Data Bus Strobe	Data Bus 64-Bit Mode
0 (MSB)	MDM[0]	MDQS[0]	MDQ[0:7]
1	MDM[1]	MDQS[1]	MDQ[8:15]
2	MDM[2]	MDQS[2]	MDQ[16:23]
3	MDM[3]	MDQS[3]	MDQ[24:31]
4	MDM[4]	MDQS[4]	MDQ[32:39]
5	MDM[5]	MDQS[5]	MDQ[40:47]
6	MDM[6]	MDQS[6]	MDQ[48:55]
7 (LSB)	MDM[7]	MDQS[7]	MDQ[56:63]

9.5.1.1 Supported DDR SDRAM Organizations

Although the DDR memory controller multiplexes row and column address bits onto 14 memory address signals and 2 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 2 logical bank-select bits and from 8–11 column address bits. Table 9-26 describes DDR SDRAM device configurations supported by the DDR memory controller.

NOTE

DDR SDRAM is limited to 27 total address bits.

Table 9-26. Supported DDR SDRAM Device Configurations

SDRAM Device	Device Configuration	Row × Column Bits	64-Bit Bank Size	Four Banks of Memory
64 Mbits	8 Mbits × 8	12 × 9	64 Mbytes	256 Mbytes
64 Mbits	4 Mbits × 16	12 × 8	32 Mbytes	128 Mbytes
128 Mbits	16 Mbits × 8	12 × 10	128 Mbytes	512 Mbytes
128 Mbits	8 Mbits × 16	12 × 9	64 Mbytes	256 Mbytes
256 Mbits	32 Mbits × 8	13 × 10	256 Mbytes	1 Gbytes
256 Mbits	16 Mbits × 16	13 × 9	128 Mbytes	512 Mbytes
512 Mbits	64 Mbits × 8	13 × 11	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits × 16	13 × 10	256 Mbytes	1 Gbytes
1 Gbits	128 Mbits × 8	14 × 11	1 Gbytes	4 Gbytes
1 Gbits	64 Mbits × 16	14 × 10	512 Mbytes	2 Gbytes

If a transaction request is issued to the DDR memory controller and the address does not lie within any of the programmed address ranges for an enabled chip select, a memory select error is flagged. Errors are described in detail in Section 9.5.12, “Error Management.”

By using a memory-polling algorithm at power-on reset or by querying the JEDEC serial presence detect capability of memory modules, system firmware uses the memory-boundary registers to configure the DDR memory controller to map the size of each bank in memory. The memory controller uses its bank map to assert the appropriate $\overline{\text{MCS}}n$ signal for memory accesses according to the provided bank starting and ending addresses. The memory banks are not required to be mapped to a contiguous address space.

9.5.2 DDR SDRAM Address Multiplexing

Table 9-27 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 msb and MA[0] as the lsb. Also, MA[10] is used as the auto precharge bit for reads and writes, so the column address can never use MA[10].

Table 9-27. 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 × 11	$\overline{\text{MRAS}}$			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	1	0												
	$\overline{\text{MCAS}}$																			11	9	8	7	6	5	4	3	2	1	0	
14 × 10	$\overline{\text{MRAS}}$				13	12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 × 11	$\overline{\text{MRAS}}$				12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	1	0												
	$\overline{\text{MCAS}}$																			11	9	8	7	6	5	4	3	2	1	0	
13 × 10	$\overline{\text{MRAS}}$					12	11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
13 × 9	$\overline{\text{MRAS}}$						12	11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					8	7	6	5	4	3	2	1	0	
12 × 10	$\overline{\text{MRAS}}$						11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		1	0											
	$\overline{\text{MCAS}}$																				9	8	7	6	5	4	3	2	1	0	
12 × 9	$\overline{\text{MRAS}}$							11	10	9	8	7	6	5	4	3	2	1	0												
	MBA																			1	0										
	$\overline{\text{MCAS}}$																					8	7	6	5	4	3	2	1	0	
12 × 8	$\overline{\text{MRAS}}$								11	10	9	8	7	6	5	4	3	2	1	0											
	MBA																				1	0									
	$\overline{\text{MCAS}}$																						7	6	5	4	3	2	1	0	

9.5.3 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 9-28) are provided by the DDR controller. All actions for these commands are described from the perspective of the SDRAM device.

- Row activate—Latches row address and initiates memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored by a precharge command before another row activate occurs.
- Precharge—Restores data from the sense amplifiers to the appropriate row. Also initializes the sense amplifiers in preparation for reading another row in the memory array, (performing another activate command). Precharge must occur after read or write, if the row address changes on the next open page mode access.
- Read—Latches column address and transfers data from the selected sense amplifier to the output buffer as determined by the column address. During each succeeding clock edge, additional data is driven without additional read commands. The amount of data transferred is determined by the burst size which defaults to 4.
- Write—Latches column address and transfers data from the data pins to the selected sense amplifier as determined by the column address. During each succeeding clock edge, additional data is transferred to the sense amplifiers from the data pins without additional write commands. The amount of data transferred is determined by the 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}, and so on). 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 9-28. SDRAM Command Table

Operation	CKE Prev.	CKE Current	$\overline{\text{MCS}}$	$\overline{\text{MRAS}}$	$\overline{\text{MCAS}}$	$\overline{\text{MWE}}$	MBA	MA10	MA
Activate	H	H	L	L	H	H	Logical bank select	Row	Row
Precharge select logical bank	H	H	L	L	H	L	Logical bank select	L	X
Precharge all logical banks	H	H	L	L	H	L	X	H	X
Read	H	H	L	H	L	H	Logical bank select	L	Column
Read with auto precharge	H	H	L	H	L	H	Logical bank select	H	Column
Write	H	H	L	H	L	L	Logical bank select	L	Column
Write with auto precharge	H	H	L	H	L	L	Logical bank select	H	Column
Mode register set	H	H	L	L	L	L	Opcode	Opcode	Opcode and mode
Auto refresh	H	H	L	L	L	H	X	X	X
Self refresh	H	L	L	L	L	H	X	X	X

9.5.4 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 if some data is irrelevant.

To accommodate available memory technologies across a wide spectrum of operating frequencies, the DDR memory controller allows the setting of the intervals defined in Table 9-29 with granularity of one memory clock cycle, except for CASLAT, which can be programmed with 1/2 clock granularity.

Table 9-29. DDR SDRAM Interface Timing Intervals

Timing Intervals	Definition
ACTTOACT	The number of clock cycles from a bank-activate command until another bank-activate command within a physical bank. This interval is listed in the AC specifications of the SDRAM.
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. The page open duration counter is reloaded with BSTOPRE each time the page is accessed (including page hits). When the counter expires, the open page is closed with a SDRAM precharge bank command as soon as possible.
CASLAT	READ latency. The number of clock cycles between the registration of a READ command by the SDRAM and the availability of the first piece of output data. If a READ command is registered at clock edge n, and the 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. 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 9.4.1.3, “DDR SDRAM Timing Configuration 1 (TIMING_CFG_1),” and Section 9.4.1.4, “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 back-to-back burst 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.

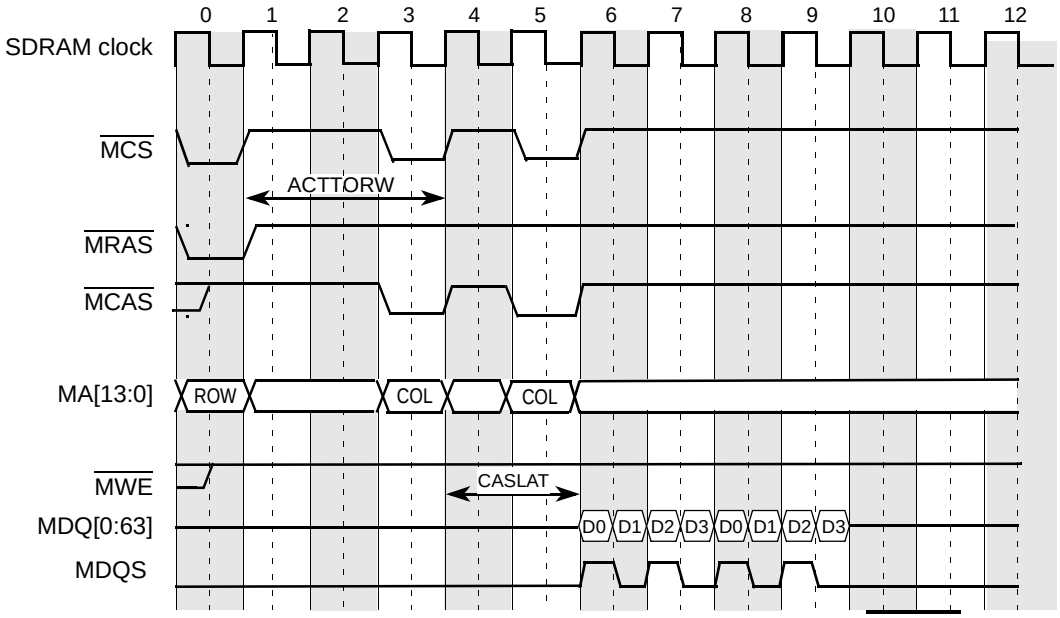


Figure 9-26. DDR SDRAM Burst Read Timing—ACTTORW = 3, MCAS Latency = 2

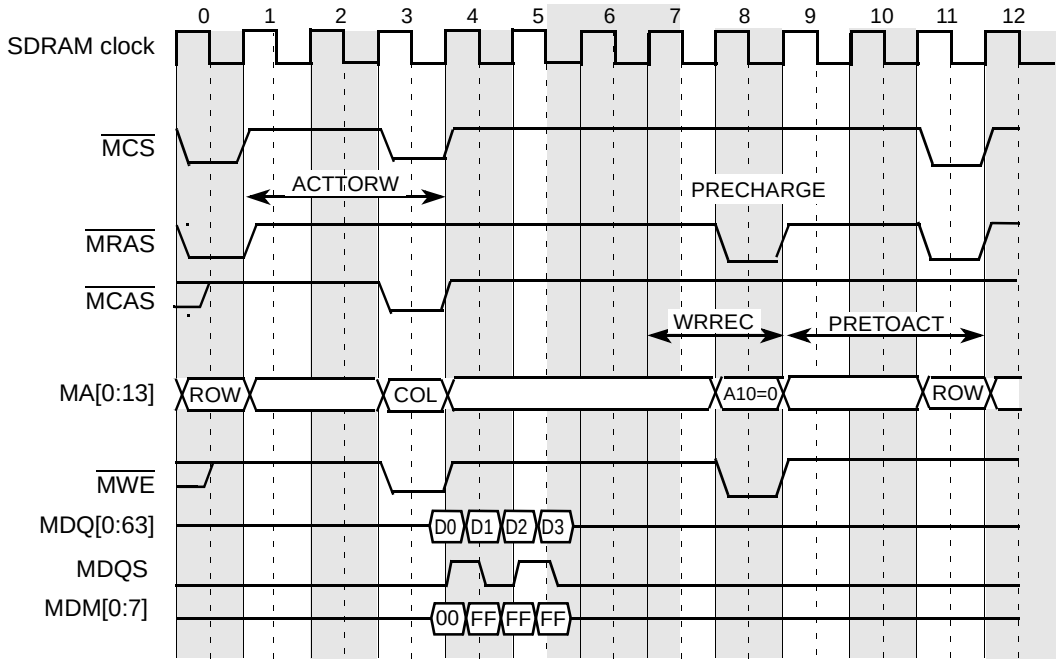


Figure 9-27. DDR SDRAM Single-Beat (Double-Word) Write Timing—ACTTORW = 3

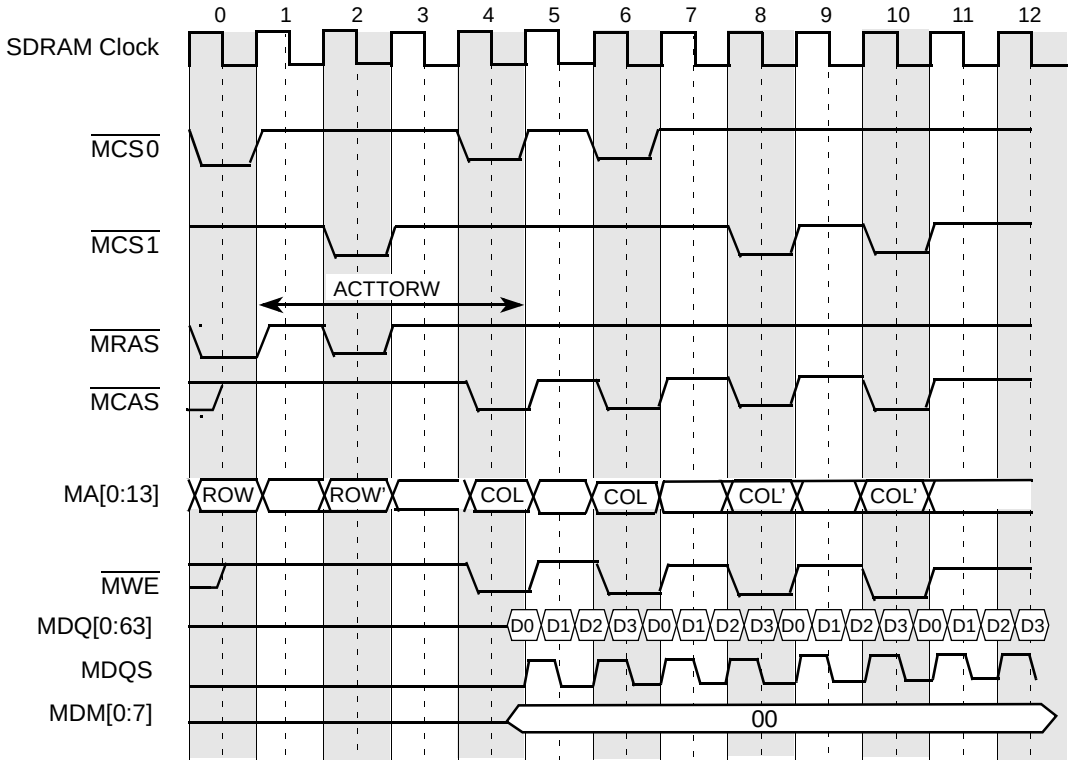


Figure 9-28. DDR SDRAM Burst Write Timing—ACTTORW = 4

9.5.4.1 Clock Distribution

- If running with many devices, zero-delay PLL clock buffers, JEDEC-JESD82 standard, should be used. These buffers were designed for DDR applications.
- 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 × 64 Mbytes DDR bank has 9-byte-wide DDR chips, resulting in 18 DDR chips in a two-bank system. In this case, each MCK/$\overline{\text{MCK}}$ signal pair should drive exactly three devices.
- PCB traces for DDR clock signals should be short, all on the same layer, and of equal length and loading as the DLL timing loop.
- DDR SDRAM manufacturers provide detailed information on PCB layout and termination issues.

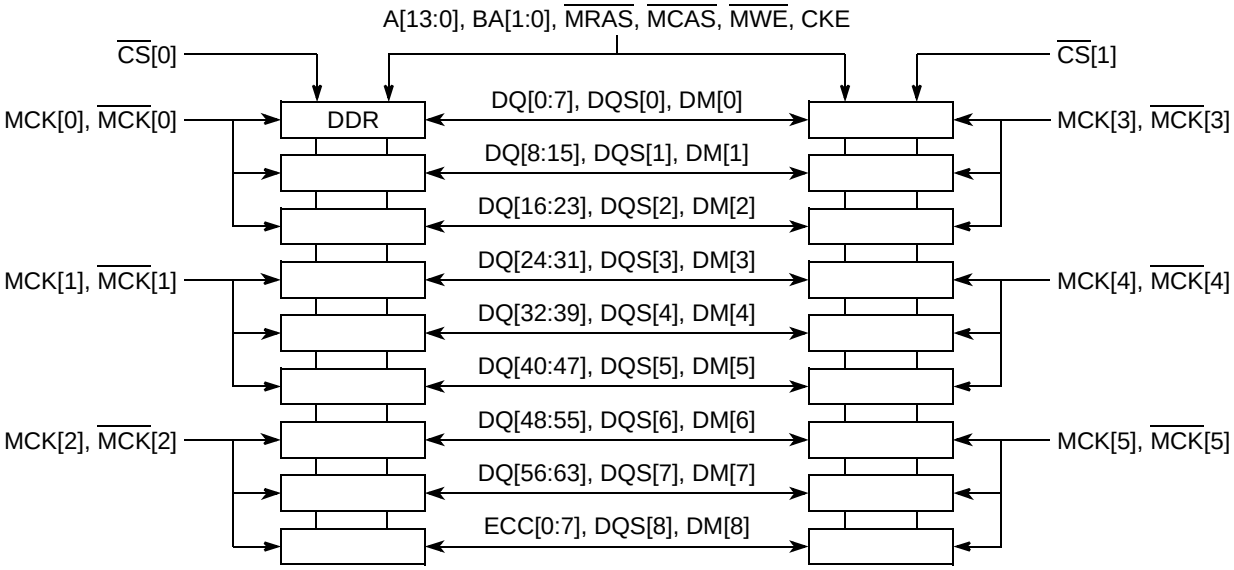


Figure 9-29. DDR SDRAM Clock Distribution Example

9.5.5 DDR SDRAM Mode-Set Command Timing

The DDR memory controller transfers the extended mode and base mode register data DDR_SDRAM_MODE[ESDMODE,SDMODE] to the SDRAM array by issuing two mode-set commands separated by two SDRAM clock periods. Figure 9-30 shows the timing of the mode-set command. The first transfer corresponds to the ESDMODE code; the second corresponds to SDMODE. Following commands must wait two SDRAM cycles.

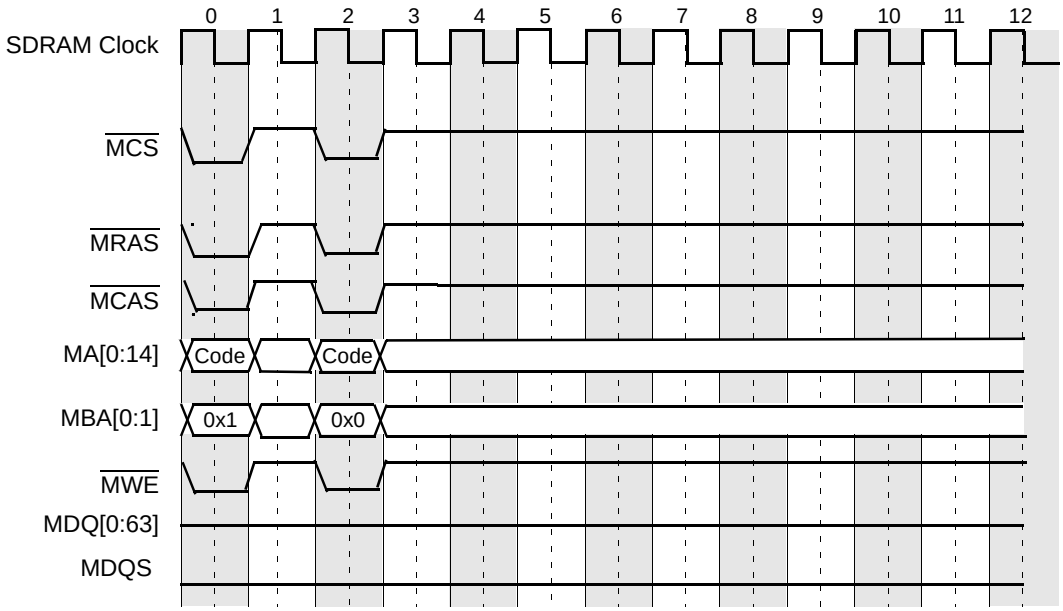


Figure 9-30. DDR SDRAM Mode-Set Command Timing

9.5.6 DDR SDRAM Registered DIMM Mode

To reduce loading, registered DIMMs latch the DDR SDRAM control signals internally before using them to access the array. Setting DDR_SDRAM_CFG[RD_EN] compensates for this delay on the DIMMs' control bus by delaying the data and data mask writes (on SDRAM buses) by an extra SDRAM clock cycle.

Enabling registered DIMM mode does not affect 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 back-to-back burst write timing.

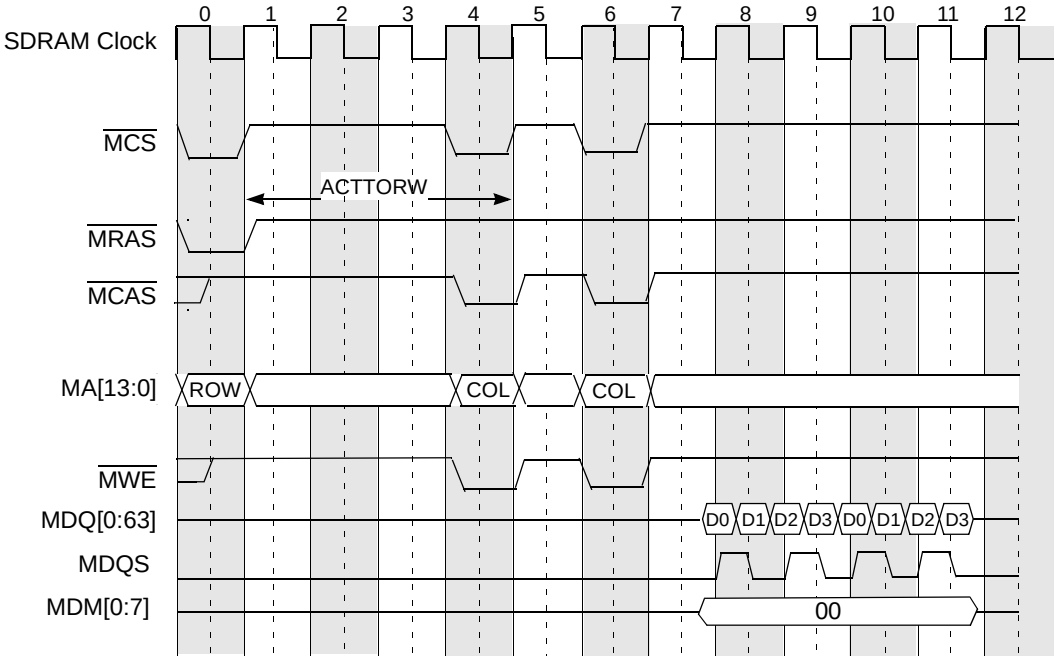


Figure 9-31. Registered DDR SDRAM DIMM Burst Write Timing

9.5.7 DDR SDRAM Write Timing Adjustments

The DDR memory controller facilitates system design flexibility by providing a write timing adjustment parameter, write data delay, (TIMING_CFG_2[WR_DATA_DELAY]) for data and DQS. The DDR SDRAM specification requires DQS be received no sooner than 75% of an SDRAM clock period—and no later than 125% of a clock period—from the capturing clock edge of the command/address at the SDRAM. The WR_DATA_DELAY parameter may be used to meet this timing requirement for a variety of system configurations, ranging from a system with one DIMM to a fully populated system with two DIMMS. TIMING_CFG_2[WR_DATA_DELAY] specifies how much to delay the launching of DQS and data from the first clock edge occurring one SDRAM clock cycle after the command is launched. The delay increment step sizes are in

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

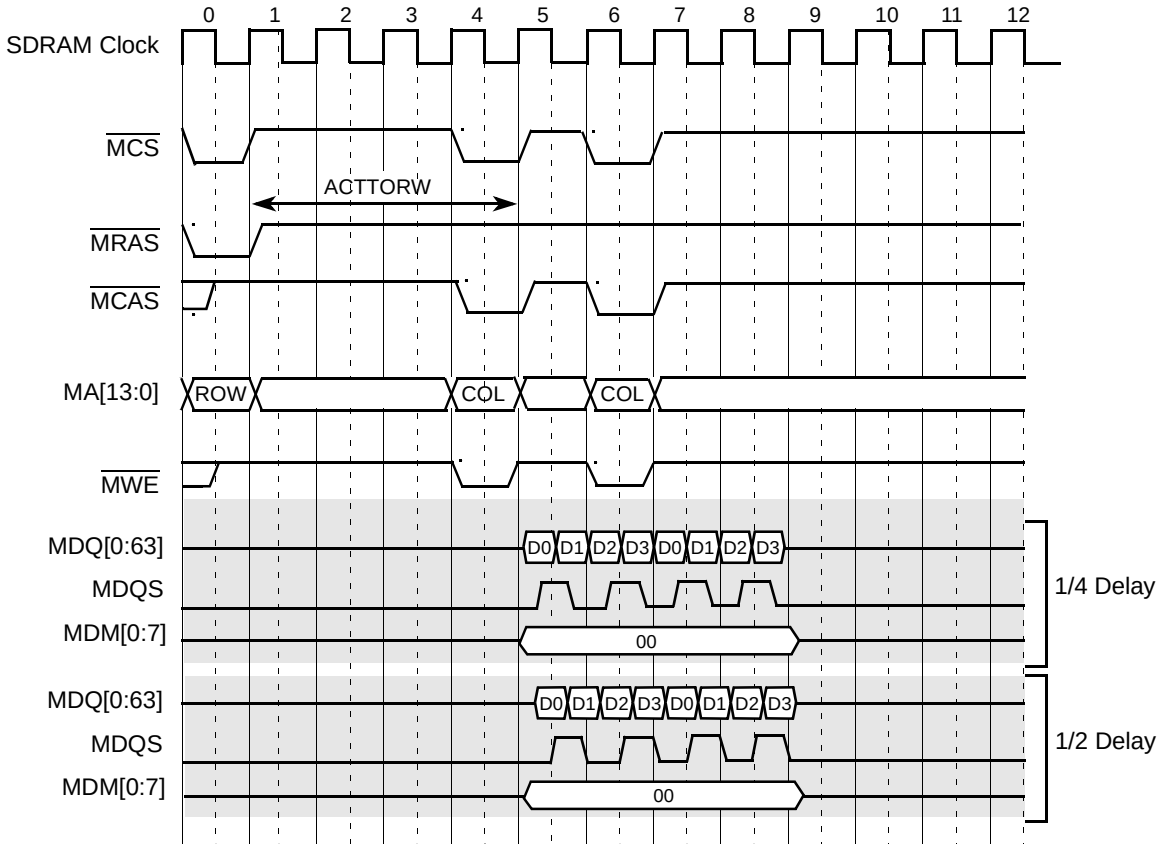


Figure 9-32. Write Timing Adjustments Example

9.5.8 DDR SDRAM Refresh

The DDR memory controller supports auto-refresh and self-refresh. Auto refresh is used during normal operation and is controlled by the DDR_SDRAM_INTERVAL[REFINT] value; self-refresh is used only when the DDR memory controller is set to enter a sleep power management state or a soft-stop state. The REFINT value, which represents the number of memory bus clock cycles between refresh cycles, must allow for possible outstanding transactions to complete before a refresh request is sent to the memory after the REFINT value is reached. If a memory transaction is in progress when the refresh interval is reached, the refresh cycle waits for the transaction to complete. In the worst case, the refresh cycle must wait the number of bus clock cycles required by the longest programmed access. To ensure that the latency caused by a memory transaction does not violate the device refresh period, it is recommended that the programmed value of REFINT be less than that required by the SDRAM.

When a refresh cycle is required, the DDR memory controller does the following:

1. Completes all current memory requests
2. Closes all open pages with a PRECHARGE-ALL command to each DDR SDRAM bank with an open page (as indicated by the row open table)
3. Issues an auto-refresh command to each DDR SDRAM bank (as identified by its chip select) to refresh one row in each logical bank of the selected physical bank

The auto-refresh commands are staggered across the four possible banks to reduce the system's instantaneous power requirements. Three sets of auto refresh commands must be issued on consecutive cycles when the memory is fully populated with two DIMMs. The initial PRECHARGE-ALL commands are also staggered in three groups for convenience. It is important to note that when entering self-refresh mode, only one refresh command is issued simultaneously to all physical banks. 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 four banks are installed. After the refresh sequence completes, any pending memory request is initiated after an inactive period specified by TIMING_CFG_1 [REFREC].

9.5.8.1 DDR SDRAM Refresh Timing

Refresh timing for the DDR SDRAM is controlled by the programmable timing parameter TIMING_CFG_1 [REFREC], which specifies the number of memory bus clock cycles from the refresh command until a logical bank activate command is allowed. The DDR memory controller implements bank staggering for refreshes, as shown in Figure 9-33 (TIMING_CFG_1 [REFREC] = 10 in this example).

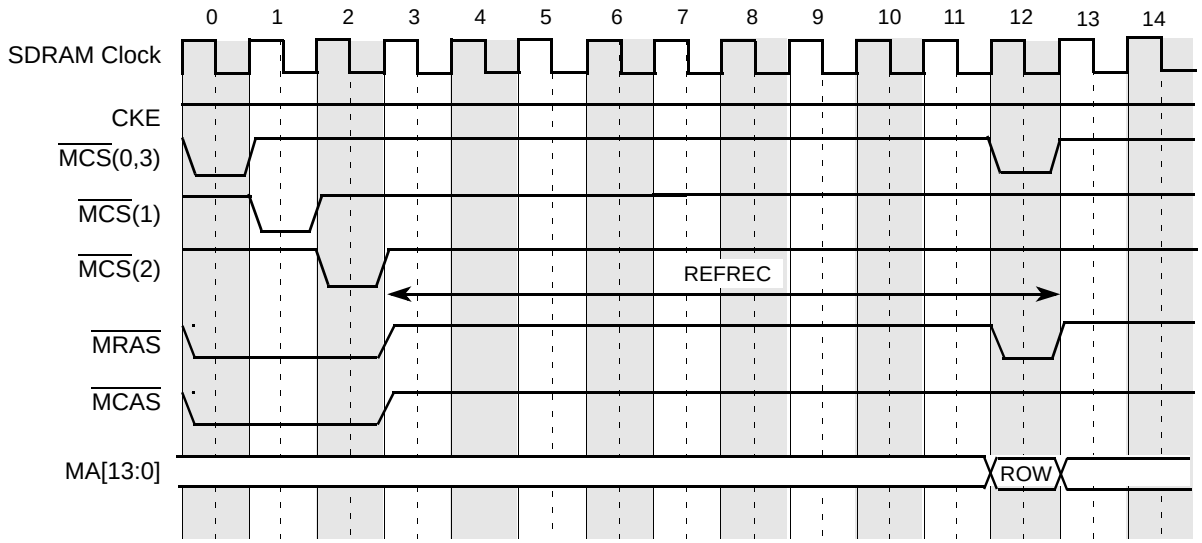


Figure 9-33. DDR SDRAM Bank-Staggered Auto-Refresh Timing

System software is responsible for optimal configuration of TIMING_CFG_1 [REFREC] at reset. Configuration must be completed before DDR SDRAM accesses are attempted.

9.5.8.2 DDR SDRAM Refresh and Power-Saving Modes

In full-on mode, the DDR memory controller supplies the normal auto refresh to SDRAM. In sleep mode, the DDR memory controller can be configured to take advantage of self-refreshing SDRAMs or to provide no refresh support. Self-refresh support is enabled with the SREN memory control parameter of the DDR_SDRAM_CFG register. Table 9-30 summarizes the refresh types available in each power-saving mode.

Table 9-30. DDR SDRAM Power-Saving Modes Refresh Configuration

Power Saving Mode	Refresh Type	SREN
Sleep	Self	1
	None	0

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.

All open pages are precharged before self refresh mode is entered.

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].

Dynamic power management mode offers tight control of the memory system's power consumption by trading power for performance through the use of 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.

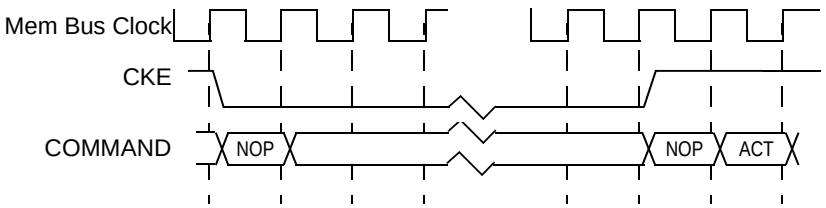


Figure 9-34. DDR SDRAM Power-Down Mode

9.5.8.2.1 Self-Refresh in Sleep Mode

The entry and exit timing for self-refreshing SDRAMs is shown in Figure 9-35 and Figure 9-36.

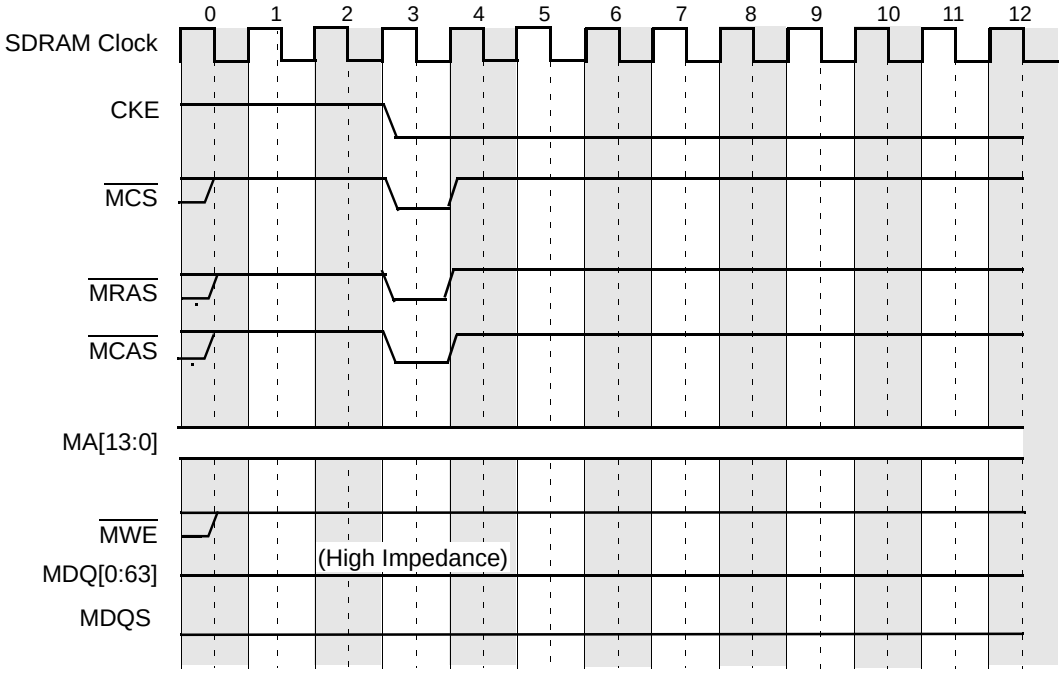


Figure 9-35. DDR SDRAM Self-Refresh Entry Timing

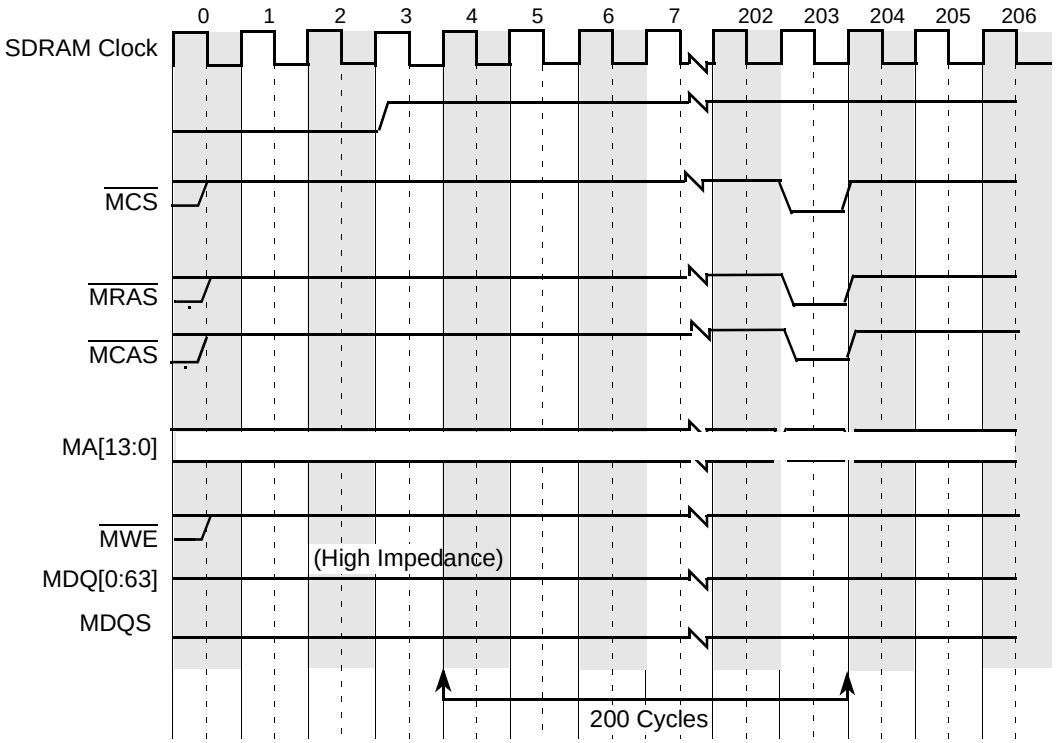


Figure 9-36. DDR SDRAM Self-Refresh Exit Timing

9.5.9 DDR Data Beat Ordering

Transfers to and from memory are always performed in four-beat bursts (four beats = 32 bytes). For transfer sizes other than four beats, the data transfers are still operated as four-beat bursts. If ECC is enabled for a sub-doubleword write transaction, a full read-modify-write is performed to properly update ECC bits. If ECC is disabled then no read-modify-write is required for sub-doubleword writes, and the data masks (MDM[0:8]) are used to prevent writing unwanted data to SDRAM. The DDR memory controller also uses data masks to prevent all unintended full double words from writing to SDRAM. For example, if a write transaction is desired with a size of one double word (8 bytes), then the 2nd, 3rd, and 4th beats of data are not written to DRAM.

Table 9-31 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 9-31. 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.

9.5.10 Page Mode and Logical Bank Retention

The DDR memory controller supports an open/closed page mode with an allowable open page for each logical bank of DRAM used. In closed page mode, the DDR memory controller uses the SDRAM AUTO PRECHARGE feature, which allows the controller to indicate that the page must be automatically closed by the DDR SDRAM after the READ or WRITE access. This is performed by using MA[10] of the address during the COMMAND phase of the access to enable AUTO PRECHARGE. AUTO PRECHARGE is non-persistent in that it is either enabled or disabled for each individual READ or WRITE command. It can however, be enabled or disabled separately for each chip select.

When the DDR memory controller operates in open page mode, it retains the currently active SDRAM page by not issuing a precharge command. The page remains opens until one of the following conditions occurs:

- Refresh interval is met
- The user-programmable DDR_SDRAM_INTERVAL[BSTOPRE] value is exceeded.
- There is a logical bank row collision with another transaction that must be issued.

Page mode can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save two to three clock cycles for subsequent burst accesses that hit in an active page. Also, better performance can be obtained by using more banks, especially in systems which use many different channels. Page mode is disabled by clearing DDR_SDRAM_INTERVAL[BSTOPRE] and CS_n_CONFIG[AP_n_EN]

9.5.11 Error Checking and Correcting (ECC)

The DDR memory controller supports error checking and correcting (ECC) for the data path between the core master and system memory. The memory detects all double-bit errors, detects all multi-bit errors within a nibble, and corrects all single-bit errors. Other errors may be detected, but are not guaranteed to be corrected or detected. Multiple-bit errors are always reported when error reporting is enabled. When a single-bit error occurs, the single-bit error counter register is incremented, and its value compared to the single-bit error trigger register. An error is reported when these values are equal. The single-bit error registers can be programmed such that minor memory faults are corrected and ignored, but a catastrophic memory failure generates an interrupt.

For writes that are smaller than 64 bits, the DDR memory controller performs a double-word read from system memory of the address for the write (checking for errors), and merges the write data with the data read from memory. Then, a new ECC code is generated for the merged double word. The data and ECC code is then written to memory. If a multi-bit error is detected on the read, the transaction completes the read-modify-write to keep the DDR memory controller from hanging. This read-modify-write operation is performed as an atomic transaction in the DDR controller. The write command is then issued 3–5 memory clocks after the completion of the read, depending on various system parameters. 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 9-32 and Table 9-33.

Table 9-32. DDR SDRAM ECC Syndrome Encoding

Data Bit	Syndrome Bit								Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
0	•	•						•	32			•	•				•
1	•		•					•	33			•		•			•
2	•			•				•	34	•		•		•			
3	•				•			•	35		•	•		•			
4	•	•				•			36			•	•		•		
5	•		•			•			37			•		•	•		
6	•			•		•			38	•		•		•	•		•
7	•				•	•			39		•	•		•	•		•
8	•	•					•		40			•	•			•	
9	•		•				•		41			•		•		•	
10	•			•			•		42	•		•		•		•	•
11	•				•		•		43		•	•		•		•	•
12	•	•				•	•	•	44			•	•		•	•	•
13	•		•			•	•	•	45			•		•	•	•	•
14	•			•		•	•	•	46	•		•		•	•	•	
15	•				•	•	•	•	47		•	•		•	•	•	
16		•	•					•	48		•				•	•	
17		•		•				•	49			•			•	•	
18		•			•			•	50				•		•	•	
19	•	•			•				51	•					•	•	
20		•	•			•			52		•				•		•
21		•		•		•			53			•			•		•
22		•			•	•			54				•		•		•
23	•	•			•	•		•	55	•					•		•
24		•	•				•		56		•					•	•
25		•		•			•		57			•				•	•
26		•			•		•		58				•			•	•
27	•	•			•		•	•	59	•						•	•
28		•	•			•	•	•	60				•	•		•	
29		•		•		•	•	•	61	•			•	•		•	•
30		•			•	•	•	•	62		•		•	•		•	•
31	•	•			•	•	•		63			•	•	•		•	•

Table 9-33. DDR SDRAM ECC Syndrome Encoding (Check Bits)

Check Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•							
1		•						
2			•					
3				•				
4					•			
5						•		
6							•	
7								•

9.5.12 Error Management

The DDR memory controller detects three different kinds of errors: single-bit, multi-bit, and memory select errors. The following discussion assumes all the relevant error detection, correction, and reporting functions are enabled as described in Section 9.4.1.16, “Memory Error Interrupt Enable (ERR_INT_EN),” Section 9.4.1.15, “Memory Error Disable (ERR_DISABLE),” and Section 9.4.1.14, “Memory Error Detect (ERR_DETECT).”

Single-bit errors are counted and reported based on the ERR_SBE value. When a single-bit error is detected, the DDR memory controller does the following:

- Corrects the data
- Increments the single-bit error counter ERR_SBE[SBEC]
- Generates an interrupt if the counter value ERR_SBE[SBEC] equals the programmable threshold ERR_SBE[SBET]
- Completes the transaction normally

If a multi-bit error is detected for a read, the DDR memory controller logs the error and generates an interrupt (if enabled, as described in Section 9.4.1.15, “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 an interrupt (if enabled, as described in Section 9.4.1.14, “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. Table 9-34 shows the errors with their descriptions.

Table 9-34. Memory Controller Errors

Category	Error	Descriptions	Action	Detect Register
Notification	Single-bit ECC threshold	The number of ECC errors has reached the threshold specified in the ERR_SBE.	The error is reported through an interrupt if enabled	The error control register only logs read versus write, not full type.
Access error	Multi-bit ECC error	A multi-bit ECC error is detected during a read, or read-modify-write memory operation.		
	Memory select error	Read, or write, address does not fall within the address range of any of the memory banks.		

9.6 Initialization/Application Information

System software must configure the DDR memory controller, using a memory polling algorithm at system start-up, to correctly map the size of each bank in memory. Then, the DDR memory controller uses its bank map to assert the appropriate $\overline{\text{MCS}}$[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 9.4.1.2, “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 Section 9.4.1, “Register Descriptions,” for more detailed descriptions of the configuration registers. These parameters are shown in Table 9-35.

Table 9-35. Memory Interface Configuration Register Initialization Parameters

Name	Description	Parameter	Section/Page
CS*n*_BNDS	Chip select memory bounds	SA*n* EA*n*	9.4.1.1/9-10
CS*n*_CONFIG	Chip select configuration	CS_*n*_EN ROW_BITS_CS_*n* COL_BITS_CS_*n*	9.4.1.2/9-10
TIMING_CFG_1	DDR SDRAM timing configuration	PRETOACT ACTTOPRE ACTTORW CASLAT REFREC WRREC ACTTOACT WRTORD WR_DATA_DELAY	9.4.1.3/9-11

Table 9-35. 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	9.4.1.5/9-14
DDR_SDRAM_MODE	DDR SDRAM mode configuration	ESDMODE SDMODE	9.4.1.6/9-16
DDR_SDRAM_INTERVAL	DDR SDRAM interval configuration	REFINT BSTOPRE	9.4.1.7/9-16

9.6.1 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 memory clocks are stable (that is, the DLL has locked or initialization is complete of all clock related configuration registers) before MEM_EN can be set, so a delay loop in the initialization code may be necessary if software is enabling the memory controller. After MEM_EN has been set, the DDR memory controller automatically performs the JEDEC-compliant initialization sequence to initialize memories according to the information in the SDMODE and ESDMODE fields of the DDR_SDRAM_MODE register. The initialization sequence 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

Note that the BA0 and BA1 bits are automatically driven appropriately during the MODE REGISTER SET commands. After this automatic initialization is complete the memory array is ready for access and the memory controller begins processing memory transactions as they arrive.

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.

10.1 Introduction

A block diagram of portions 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 the following sources:

- External interrupts. Triggered by signals external to the integrated device, namely IRQ[0:11]
- Internal interrupts. Triggered by signal internal to the integrated device, typically representing major blocks by 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 (TSECs), the CPM, the performance monitor, and the I^2C controller.
- Interrupts generated from within the PIC itself. Names the messaging, global timer, and interprocessor interrupts defined by the OpenPIC specification

The PIC can be configured such that interrupts can be directed as follows:

- To the core through the external interrupt signal, *int*. For interrupts signalled through the *int* signal, the PIC unit prioritizes and manages interrupts such that the highest priority interrupt is always recognized and taken, and that any lower priority interrupt that is deferred when a higher priority interrupt is taken, resumes execution as soon as all higher priority interrupts have been handled.
- To the core through the critical interrupt signal, *cint*
- To an external interrupt controller, through the $\overline{\text{IRQ\_OUT}}$ signal

Table 10-1 shows which signalling mechanisms are available to each interrupt source.

Table 10-1. Interrupt Source Signalling Options

Source	*int*	*cint*	$\overline{\text{IRQ\_OUT}}$
Internal	√	√	√
External	√	√	√
Message	√	√	√
Interprocessor	√	x	x
Global timer	√	x	x

Note that the PIC can be disabled, in which case internal interrupts are passed through $\overline{\text{IRQ\_OUT}}$. All other interrupts are disabled.

10.1.1 Overview

The 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.

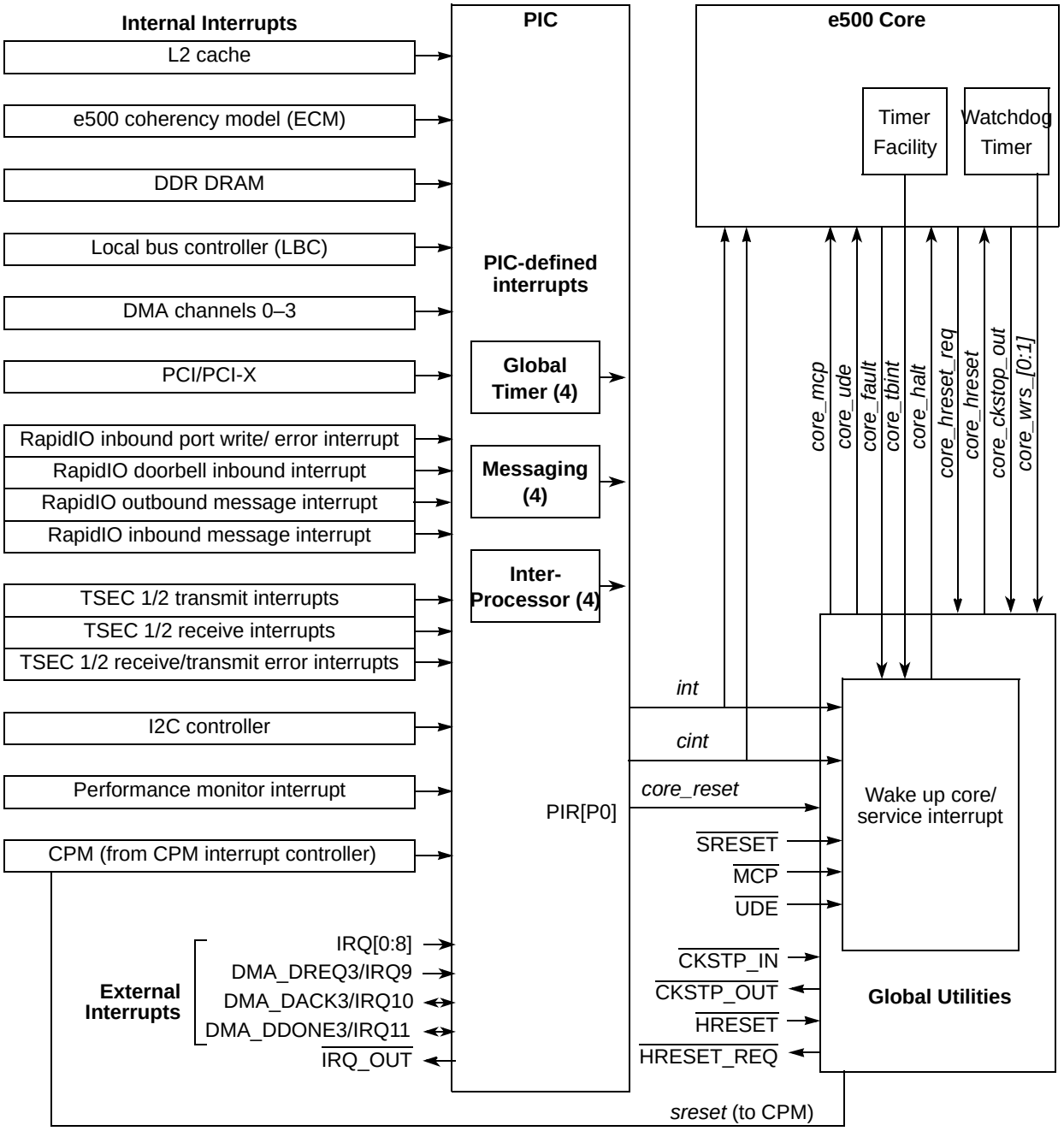


Figure 10-1. MPC8560 Interrupt Sources Block Diagram

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 that can be clocked with the CCB (platform) clock or the RTC input.
- 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 Interrupts to the Processor Core

The *int* signal, which causes the external interrupt exception, is the main interrupt output from the PIC unit to the processor core. External, internal, and message interrupts can alternately be configured as critical interrupts (in the destination registers); these are reported to the core through the *cint* signal and do not use the PIC logic described in Section 10.4.1, "Flow of Interrupt Control."

The Book E architecture implemented by the e500 core defines a separate critical interrupt type with its own save and restore registers (CSRR0 and CSRR1) and return instruction (Return from Critical Interrupt, **rfci**). In addition to the external and critical interrupts that are generated by the PIC, other MPC8560 conditions (shown in Table 10-2) cause interrupts to the core (and wake up the core when it is in a low-power state). Note that interrupts from port C of the CPM are a special case and do not reach the PIC when the device is asleep. Therefore, they do not cause the device to wake up.

Table 10-2. Processor Interrupts Generated Outside the Core—Types and Sources

Core Interrupt Type	Signaled by (Input to Core)	Sources
PIC-Programmable Interrupts		
External interrupt	*int*	Generated by the PIC, as described in Section 10.1.5, "Interrupt Sources."
Critical interrupt	*cint*	Generated by the PIC, as described in Section 10.1.5, "Interrupt Sources."
Other Interrupts Generated Outside the Core		
Machine check	*core_mcp* (MPC8560 causes)	• $\overline{\text{MCP}}$ • $\overline{\text{SRESET}}$ • Assertion of *core_mcp* by global utilities block

Table 10-2. Processor Interrupts Generated Outside the Core—Types and Sources (continued)

Core Interrupt Type	Signaled by (Input to Core)	Sources
Unconditional debug event	*core_ude*	$\overline{\text{UDE}}$. Asserting $\overline{\text{UDE}}$ generates an unconditional debug exception type debug interrupt and sets a bit in the debug status register, DBSR[UDE], as described in Section 6.13.2, “Debug Status Register (DBSR).”
Reset	*core_hreset*	• $\overline{\text{HRESET}}$ assertion (and negation) • *core_hreset_req*. Output from core—caused by writing to the core DBCR0[RST]. This condition is additionally qualified with MSR[DE] and DBCR0[IDM] bits. Note that assertion of this signal causes a hard reset of the core only. • *core_hreset_req* can also be caused by a second timer timeout condition as described in Section 6.6.1, “Timer Control Register (TCR).” • *core_reset*. Output from PIC. See Section 10.3.1.4, “Processor Initialization Register (PIR).”

The global utilities block monitors two additional interrupt conditions generated by the e500 core (*core_tbint* and *core_fault_out* signals), as shown in Figure 10-1. Assertion of either of these signals causes the processor to exit a low-power state. These cases are caused by core conditions, and after the global utilities logic wakes up the core, they are handled by the core as shown in Table 10-3.

Table 10-3. e500 Core-Generated Interrupts that Cause a Wake-up

Core Interrupt Type	Signaled by (Output from Core)	Sources
Fixed interval timer	*core_tbint*	The source of both of these interrupts is the time base facility within the e500 core. The MPC8560 monitors this core output signal and considers it a processor interrupt for the purposes of power management (causes the core to exit a low-power state). For more information about the interaction between core-generated signals and power management, see Chapter 17, “Global Utilities.”
Decrementer		
Machine check	*core_fault_out*	Occurs when the L1 cache has a parity error on a snoop push operation, which can occur while the core is halted. The MPC8560 monitors this signal and considers it a processor interrupt for the purposes of power management (causes the core to exit a low-power state).

10.1.4 Modes of Operation

Mixed or pass-through mode can be chosen by setting or clearing GCR[M] as described in Section 10.3.1.2, “Global Configuration Register (GCR).”

10.1.4.1 Mixed Mode (GCR[M] = 1)

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).”

10.1.4.2 Pass-Through Mode (GCR[M] = 0)

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 pass-through mode, in which active-high interrupts from external source IRQ0 are passed directly to the e500 core, as shown in Table 10-2, all other external interrupt signals are ignored. Thus, the interrupt signal from an external interrupt controller can be connected to IRQ0 and cause direct interrupts to the processor. The PIC does not perform a vector fetch from an 8259 interrupt controller.

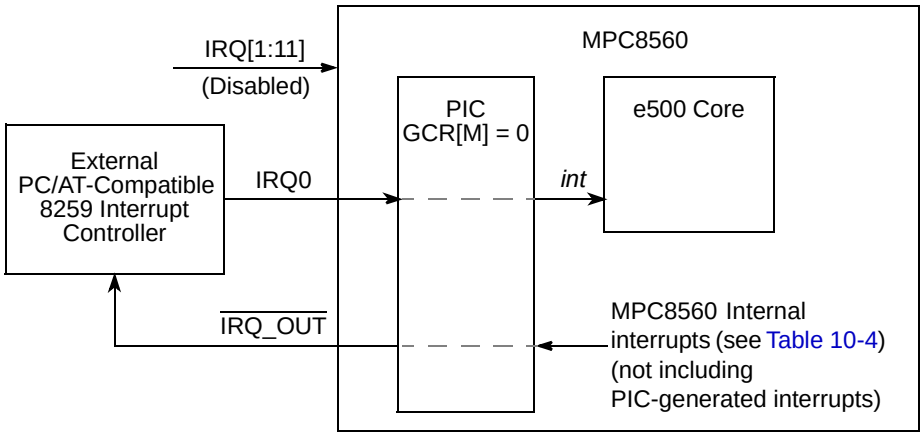


Figure 10-2. Pass-Through Mode Example

When pass-through mode is enabled, the internally-generated interrupts shown in Table 10-4 are not forwarded to the e500 core. Instead, the PIC passes the raw interrupts from the internal sources to $\overline{\text{IRQ\_OUT}}$.

Note that in pass-through mode, interrupts generated by the PIC itself (global timers, interprocessor, and message register interrupts) cannot be used. If internal or PIC-generated interrupts must be reported internally to the processor, pass-through mode must be disabled.

10.1.5 Interrupt Sources

Aside from the sources of machine check, unconditional debug event, and reset interrupts to the core described in Table 10-2, 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, local bus controller, 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. Used only 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

10.1.5.1 Interrupt Routing—Mixed Mode

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 (if the incoming interrupt has its xIDR[EP] bit set). Alternatively, 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. As a third alternative (if neither xIDR[CI] or xIDR[EP] are set), the interrupt can be serviced as a normal external interrupt by the processor core (through the *int* signal). In this case, the interrupt is latched by the interrupt pending register (IPR) and the interrupt flow described in Section 10.4.1, “Flow of Interrupt Control,” is followed.

10.1.5.2 Internal Interrupt Sources

Table 10-4 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 10-4. Internal Interrupt Assignments

Internal Interrupt Number	Interrupt Source	Internal Interrupt Number	Interrupt Source
0	L2 cache	14	TSEC 1 receive interrupt
1	ECM	15–17	Reserved
2	DDR DRAM	18	TSEC 1 receive/transmit error interrupt
3	LBC	19	TSEC 2 transmit interrupt
4	DMA channel 0	20	TSEC 2 receive interrupt
5	DMA channel 1	21–23	Reserved
6	DMA channel 2	24	TSEC 2 receive/transmit error interrupt
7	DMA channel 3	25	unused
8	PCI/PCI-X	26	unused
9	RapidIO inbound port write/ error interrupt	27	I^2C controller
10	RapidIO doorbell inbound interrupt	28	Performance monitor interrupt
11	RapidIO outbound message interrupt	29	Unused
12	RapidIO inbound message interrupt	30	CPM (note that interrupts from port C are not signaled to the PIC when the device is asleep)
13	TSEC 1 transmit interrupt	31	Unused

10.2 External Signal Descriptions

The following sections provide an overview and detailed descriptions of the PIC signals.

10.2.1 Signal Overview

PIC interface signals are described in Table 10-5. There are 12 distinct external interrupt request input signals (IRQ[0:11]) and 1 interrupt request output signal ($\overline{\text{IRQ\_OUT}}$). As Table 10-5 shows, three IRQ inputs are multiplexed with DMA signals for DMA channel 3.

Table 10-5. 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	Processor machine check
$\overline{\text{UDE}}$	I	Unconditional debug event

1 IRQ9–IRQ11 are multiplexed with DMA3 signals. These functions are mutually exclusive; the active function is specified in PMUXCR of the global utilities block as described in Section 17.4.1.10, “Alternate Function Signal Multiplex Control Register (PMUXCR).”

10.2.2 Detailed Signal Descriptions

Table 10-6 provides detailed descriptions of the external PIC signals.

Table 10-6. 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.

Table 10-6. Interrupt Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
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 10.1.4, "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 IRQ_OUT.
		Timing	Because external interrupts are asynchronous with respect to the system clock, both assertion and negation of IRQ_OUT occurs asynchronously with respect to the interrupt source. All timing given here is approximate. Assertion—Internal interrupt source: 2 CCB clock cycles after interrupt occurs. External interrupt source: 4 cycles after interrupt occurs. Message interrupts: 2 cycles after write to message register. Negation—Follows interrupt source negation with the following delay: Internal interrupt: 2 CCB clock cycles External interrupt: 4 cycles. Message interrupts: 2 cycles after message register cleared.
MCP	I	Machine check processor. Assertion causes a machine check interrupt to the e500 core. Note that if the e500 core is not configured to process machine check interrupts (MSR[ME] = 0), assertion of MCP causes a checkstop condition. Note that internal sources for the internal *core_mcp* signal can also cause a machine check interrupt to the processor core, as described in Section 17.4.1.13, "Machine Check Summary Register (MCPSUMR)," Table 10-2 and Table 10-3.	
		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—Because MCP is edge-triggered, it can be negated one clock after its assertion.
UDE	I	Unconditional debug event. Assertion signal causes an unconditional debug exception to the e500 core.	
		State Meaning	Asserted—Indicates that the MPC8560 should initiate an unconditional debug event interrupt to the processor core. Negated—Indicates that unconditional debug event handling is not being requested by UDE.
		Timing	Assertion—May occur at any time, asynchronous to any clock. Negation—Should remain asserted until software in the unconditional debug event interrupt handler causes the external device asserting the UDE signal to negate it.

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 10-7, 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 10-7. PIC Register Address Map

Offset	Register	Access	Reset	Section/Page
PIC Register Address Map—Global Registers				
0x4_0000–0x4_0030	Reserved	—	—	—
0x4_0040	IPIDR0—Interprocessor interrupt 0 (IPI 0) dispatch register [1]	W	0x0000_0000	10.3.7.1/10-39
0x4_0050	IPIDR1—IPI 1 dispatch register			
0x4_0060	IPIDR2—IPI 2 dispatch register			
0x4_0070	IPIDR3—IPI 3 dispatch register			
0x4_0080	CTPR—Current task priority register	R/W	0x0000_000F	10.3.7.2/10-40
0x4_0090	WHOAMI—Who am I register	R	0x0000_0000	10.3.7.3/10-41
0x4_00A0	IACK—Interrupt acknowledge register	R	0x0000_0000	10.3.7.4/10-41
0x4_00B0	EOI—End of interrupt register	W	0x0000_0000	10.3.7.5/10-42
0x4_00C0–0x4_0FF0	Reserved	—	—	—
0x4_1000	FRR—Feature reporting register	R	0x0037_0002	10.3.1.1/10-16
0x4_1010	Reserved	—	—	—
0x4_1020	GCR—Global configuration register	R/W	0x0000_0000	10.3.1.2/10-17
0x4_1030	Reserved	—	—	—
0x4_1040–0x4_1070	Vendor reserved	—	—	—
0x4_1080	VIR—Vendor identification register	R	0x0000_0000	10.3.1.3/10-17
0x4_1090	PIR—Processor initialization register	R/W	0x0000_0000	10.3.1.4/10-18
0x4_10A0	IPIVPR0—IPI 0 vector/priority register	R/W	0x8000_0000	10.3.1.5/10-19
0x4_10B0	IPIVPR1—IPI 1 vector/priority register			
0x4_10C0	IPIVPR2—IPI 2 vector/priority register			
0x4_10D0	IPIVPR3—IPI 3 vector/priority register			
0x4_10E0	SVR—Spurious vector register	R/W	0x0000_FFFF	10.3.1.6/10-19
0x4_10F0	TFRR—Timer frequency reporting register	R/W	0x0000_0000	10.3.2.1/10-20
0x4_1100	GTCCR0—Global timer 0 current count register	R	0x0000_0000	10.3.2.2/10-21
0x4_1110	GTBCR0—Global timer 0 base count register	R/W	0x8000_0000	10.3.2.3/10-21

Table 10-7. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1120	GTVPR0—Global timer 0 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-22
0x4_1130	GTDR0—Global timer 0 destination register	R/W	0x0000_0001	10.3.2.5/10-23
0x4_1140	GTCCR1—Global timer 1 current count register	R	0x0000_0000	10.3.2.2/10-21
0x4_1150	GTBCR1—Global timer 1 base count register	R/W	0x8000_0000	10.3.2.3/10-21
0x4_1160	GTVPR1—Global timer 1 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-22
0x4_1170	GTDR1—Global timer 1 destination register	R/W	0x0000_0001	10.3.2.5/10-23
0x4_1180	GTCCR2—Global timer 2 current count register	R	0x0000_0000	10.3.2.2/10-21
0x4_1190	GTBCR2—Global timer 2 base count register	R/W	0x8000_0000	10.3.2.3/10-21
0x4_11A0	GTVPR2—Global timer 2 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-22
0x4_11B0	GTDR2—Global timer 2 destination register	R/W	0x0000_0001	10.3.2.5/10-23
0x4_11C0	GTCCR3—Global timer 3 current count register	R	0x0000_0000	10.3.2.2/10-21
0x4_11D0	GTBCR3—Global timer 3 base count register	R/W	0x8000_0000	10.3.2.3/10-21
0x4_11E0	GTVPR3—Global timer 3 vector/priority register	R/W	0x8000_0000	10.3.2.4/10-22
0x4_11F0	GTDR3—Global timer 3 destination register	R/W	0x0000_0001	10.3.2.5/10-23
0x4_1200–0x4_12F0	Reserved	—	—	—
0x4_1300	TCR—Timer control register	R/W	0x0000_0000	10.3.2.6/10-23
0x4_1310	IRQSR0—$\overline{\text{IRQ\_OUT}}$ summary register 0	R	0x0000_0000	10.3.3.1/10-26
0x4_1320	IRQSR1—$\overline{\text{IRQ\_OUT}}$ summary register 1	R	0x0000_0000	10.3.3.2/10-27
0x4_1330	CISR0—Critical Interrupt summary register 0	R	0x0000_0000	10.3.3.3/10-27
0x4_1340	CISR1—Critical Interrupt summary register 1	R	0x0000_0000	10.3.3.4/10-28
0x4_1350	PM0MR0—Performance monitor 0 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-29
0x4_1360	PM0MR1—Performance monitor 0 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-29
0x4_1370	PM1MR0—Performance monitor 1 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-29
0x4_1380	PM1MR1—Performance monitor 1 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-29
0x4_1390	PM2MR0—Performance monitor 2 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-29
0x4_13A0	PM2MR1—Performance monitor 2 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-29
0x4_13B0	PM3MR0—Performance monitor 3 mask register 0	R/W	0x00FF_FFFF	10.3.4.1/10-29
0x4_13C0	PM3MR1—Performance monitor 3 mask register 1	R/W	0xFFFF_FFFF	10.3.4.2/10-29
0x4_13D0–0x4_13F0	Reserved	—	—	—

Table 10-7. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1400	MSGR0—Message register 0	R/W	0x0000_0000	10.3.5.1/10-30
0x4_1410	MSGR1—Message register 1			
0x4_1420	MSGR2—Message register 2			
0x4_1430	MSGR3—Message register 3			
0x4_1440–0x4_14F0	Reserved	—	—	—
0x4_1500	MER—Message enable register	R/W	0x0000_0000	10.3.5.2/10-30
0x4_1510	MSR—Message status register	R/W	0x0000_0000	10.3.5.3/10-31
0x4_1520–0x4_FFF0	Reserved	—	—	—
PIC Register Address Map—Interrupt Source Configuration Registers				
0x5_0000	EIVPR0—External interrupt 0 (IRQ0) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0010	EIDR0—External interrupt 0 (IRQ0) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0020	EIVPR1—External interrupt 1 (IRQ1) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0030	EIDR1—External interrupt 1 (IRQ1) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0040	EIVPR2—External interrupt 2 (IRQ2) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0050	EIDR2—External interrupt 2 (IRQ2) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0060	EIVPR3—External interrupt 3 (IRQ3) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0070	EIDR3—External interrupt 3 (IRQ3) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0080	EIVPR4—External interrupt 4 (IRQ4) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0090	EIDR4—External interrupt 4 (IRQ4) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_00A0	EIVPR5—External interrupt 5 (IRQ5) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_00B0	EIDR5—External interrupt 5 (IRQ5) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_00C0	EIVPR6—External interrupt 6 (IRQ6) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_00D0	EIDR6—External interrupt 6 (IRQ6) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_00E0	EIVPR7—External interrupt 7 (IRQ7) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_00F0	EIDR7—External interrupt 7 (IRQ7) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0100	EIVPR8—External interrupt 8 (IRQ8) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0110	EIDR8—External interrupt 8 (IRQ8) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0120	EIVPR9—External interrupt 9 (IRQ9) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0130	EIDR9—External interrupt 9 (IRQ9) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0140	EIVPR10—External interrupt 10 (IRQ10) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32

Table 10-7. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_0150	EIDR10—External interrupt 10 (IRQ10) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0160	EIVPR11—External interrupt 11 (IRQ11) vector/priority register	R/W	0x8000_0000	10.3.6.1/10-32
0x5_0170	EIDR11—External interrupt 11 (IRQ11) destination register	R/W	0x0000_0001	10.3.6.2/10-33
0x5_0180–0x5_01F0	Reserved	—	—	—
0x5_0200	IIVPR0—Internal interrupt 0 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0210	IIDR0—Internal interrupt 0 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0220	IIVPR1—Internal interrupt 1 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0230	IIDR1—Internal interrupt 1 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0240	IIVPR2—Internal interrupt 2 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0250	IIDR2—Internal interrupt 2 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0260	IIVPR3—Internal interrupt 3 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0270	IIDR3—Internal interrupt 3 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0280	IIVPR4—Internal interrupt 4 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0290	IIDR4—Internal interrupt 4 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_02A0	IIVPR5—Internal interrupt 5 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_02B0	IIDR5—Internal interrupt 5 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_02C0	IIVPR6—Internal interrupt 6 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_02D0	IIDR6—Internal interrupt 6 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_02E0	IIVPR7—Internal interrupt 7 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_02F0	IIDR7—Internal interrupt 7 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0300	IIVPR8—Internal interrupt 8 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0310	IIDR8—Internal interrupt 8 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0320	IIVPR9—Internal interrupt 9 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0330	IIDR9—Internal interrupt 9 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0340	IIVPR10—Internal interrupt 10 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0350	IIDR10—Internal interrupt 10 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0360	IIVPR11—Internal interrupt 11 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0370	IIDR11—Internal interrupt 11 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0380	IIVPR12—Internal interrupt 12 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0390	IIDR12—Internal interrupt 12 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_03A0	IIVPR13—Internal interrupt 13 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34

Table 10-7. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_03B0	IIDR13—Internal interrupt 13 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_03C0	IIVPR14—Internal interrupt 14 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_03D0	IIDR14—Internal interrupt 14 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_03E0	IIVPR15—Internal interrupt 15 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_03F0	IIDR15—Internal interrupt 15 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0400	IIVPR16—Internal interrupt 16 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0410	IIDR16—Internal interrupt 16 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0420	IIVPR17—Internal interrupt 17 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0430	IIDR17—Internal interrupt 17 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0440	IIVPR18—Internal interrupt 18 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0450	IIDR18—Internal interrupt 18 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0460	IIVPR19—Internal interrupt 19 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0470	IIDR19—Internal interrupt 19 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0480	IIVPR20—Internal interrupt 20 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0490	IIDR20—Internal interrupt 20 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_04A0	IIVPR21—Internal interrupt 21 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_04B0	IIDR21—Internal interrupt 21 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_04C0	IIVPR22—Internal interrupt 22 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_04D0	IIDR22—Internal interrupt 22 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_04E0	IIVPR23—Internal interrupt 23 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_04F0	IIDR23—Internal interrupt 23 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0500	IIVPR24—Internal interrupt 24 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0510	IIDR24—Internal interrupt 24 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0520	IIVPR25—Internal interrupt 25 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0530	IIDR25—Internal interrupt 25 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0540	IIVPR26—Internal interrupt 26 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0550	IIDR126—Internal interrupt 26 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0560	IIVPR27—Internal interrupt 27 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0570	IIDR27—Internal interrupt 27 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0580	IIVPR28—Internal interrupt 28 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_0590	IIDR28—Internal interrupt 28 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_05A0	IIVPR29—Internal interrupt 29 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34

Table 10-7. PIC Register Address Map (continued)

Offset	Register	Access	Reset	Section/Page
0x5_05B0	IIDR29—Internal interrupt 29 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_05C0	IIVPR30—Internal interrupt 30 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_05D0	IIDR30—Internal interrupt 30 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_05E0	IIVPR31—Internal interrupt 31 vector/priority register	R/W	0x8080_0000	10.3.6.3/10-34
0x5_05F0	IIDR31—Internal interrupt 31 destination register	R/W	0x0000_0001	10.3.6.4/10-35
0x5_0600–0x5_15F0	Reserved	—	—	—
0x5_1600	MIVPR0—Messaging interrupt 0 (MSG 0) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-36
0x5_1610	MIDR0—Messaging interrupt 0 (MSG 0) destination register	R/W	0x0000_0001	10.3.6.6/10-37
0x5_1620	MIVPR1—Messaging interrupt 1 (MSG 1) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-36
0x5_1630	MIDR1—Messaging interrupt 1 (MSG 1) destination register	R/W	0x0000_0001	10.3.6.6/10-37
0x5_1640	MIVPR2—Messaging interrupt 2 (MSG 2) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-36
0x5_1650	MIDR2—Messaging interrupt 2 (MSG 2) destination register	R/W	0x0000_0001	10.3.6.6/10-37
0x5_1660	MIVPR3—Messaging interrupt 3 (MSG 3) vector/priority register	R/W	0x8000_0000	10.3.6.5/10-36
0x5_1670	MIDR3—Messaging interrupt 3 (MSG 3) destination register	R/W	0x0000_0001	10.3.6.6/10-37
0x5_1680–0x5_FFF0	Reserved	—	—	—
PIC Register Address Map—Per-CPU Registers				
0x6_0000–0x6_0030	Reserved	—	—	—
0x6_0040	IPIDR0—P0 IPI 0 dispatch register	W	All zeros	10.3.7.1/10-39
0x6_0050	IPIDR1—P0 IPI 1 dispatch register			
0x6_0060	IPIDR2—P0 IPI 2 dispatch register			
0x6_0070	IPIDR3—P0 IPI 3 dispatch register			
0x6_0080	CTPR0—P0 current task priority register	R/W	0x0000_000F	10.3.7.2/10-40
0x6_0090	WHOAMI0—P0 who am I register	R	All zeros	10.3.7.3/10-41
0x6_00A0	IACK0—P0 interrupt acknowledge register	R	All zeros	10.3.7.4/10-41
0x6_00B0	EOI0—P0 end of interrupt register	W	All zeros	10.3.7.5/10-42

[1] Note that these registers provider private access space to the same set of registers at offset 0x6_xxxx. This private access is described in Section 10.3.7, “Per-CPU Registers.”

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 its 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 10-31, 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 per-CPU registers, see Section 10.3.7, "Per-CPU Registers."

NOTE

Register fields designated as write-one-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

10.3.1.1 Feature Reporting Register (FRR)

The feature reporting register (FRR) shown in Figure 10-3 provides information about interrupt and processor configurations. It also informs the programming environment of the controller version.

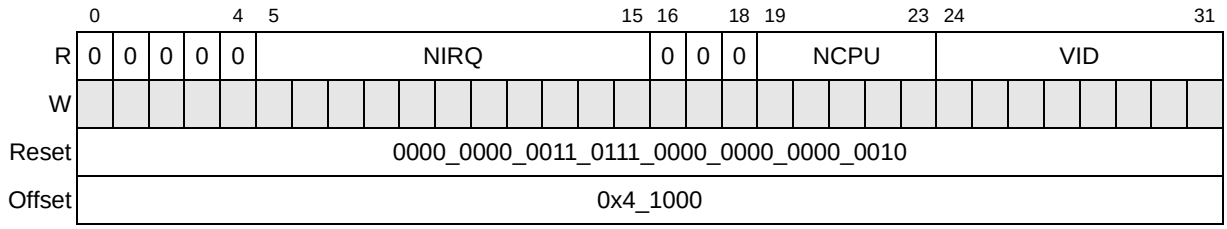


Figure 10-3. Feature Reporting Register (FRR)

Table 10-8 describes the FRR fields.

Table 10-8. FRR Field Descriptions

Bits	Name	Description
0–4	—	Reserved.
5–15	NIRQ	Number of interrupts. Contains the binary value of the maximum number of interrupt sources supported minus one. 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.
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 OpenPIC 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 10-4 controls the PIC's operating mode, and allows software to reset the PIC.

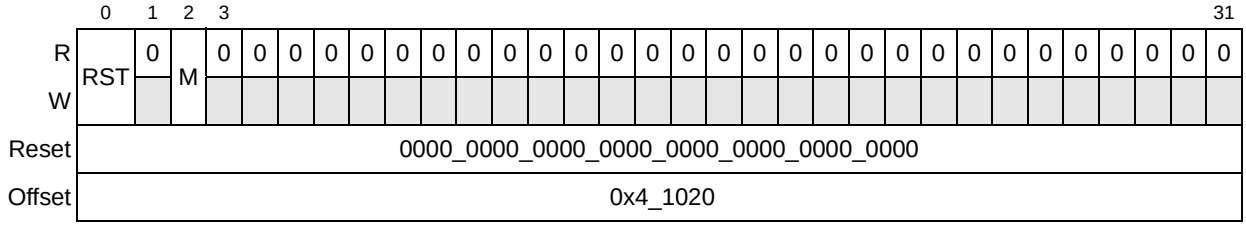


Figure 10-4. Global Configuration Register (GCR)

Table 10-9 describes the GCR fields.

Table 10-9. 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 10.4.7, "Reset of the PIC," for more information on resetting the PIC.
1	—	Reserved
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 10.1.4, "Modes of Operation," for more details. 1 Mixed mode. Interrupts are handled by the normal priority and delivery mechanisms of the PIC. See Section 10.1.4, "Modes of Operation," for more details.
3–31	—	Reserved

10.3.1.3 Vendor Identification Register (VIR)

The vendor identification register (VIR) shown in Figure 10-5 has specific read-only information about the vendor and the device revision.

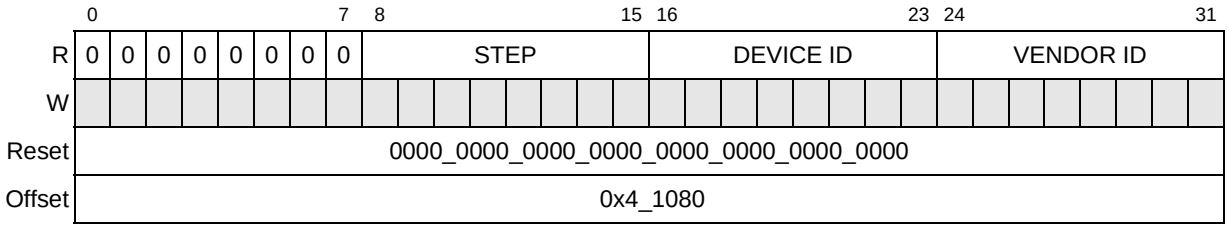


Figure 10-5. Vendor Identification Register (VIR)

Table 10-10 describes the VIR fields.

Table 10-10. VIR Field Descriptions

Bits	Name	Description
0–7	—	Reserved
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 in the PIC provides a mechanism for software to reset the processor. The *core_reset* signal to the processor is held active until a zero is written to the processor initialization register field. Thus, PIR should only be written by an external host and the external host should wait at least 100 µsec to clear this field after writing it. The processor should not attempt to write to PIR because writing to it resets the core, and the core will not be able to clear the reset field, and so it would remain in reset indefinitely.

Note that although the architecture was designed to support multiple processing cores, only fields corresponding to processors on the device are implemented, as shown in Figure 10-6. In a uniprocessor system, only PIR[P0] is implemented.

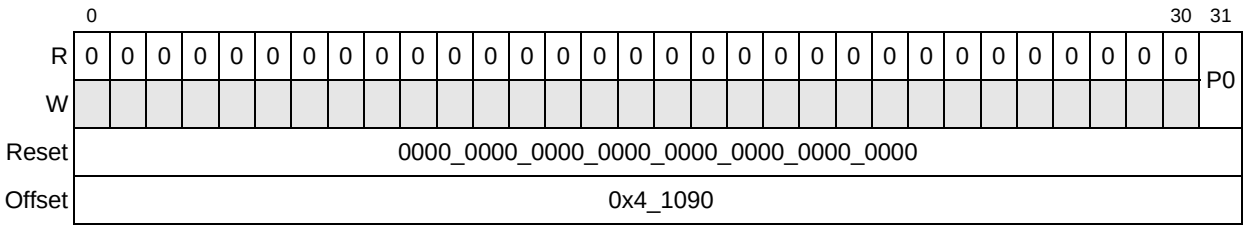


Figure 10-6. Processor Initialization Register (PIR)

Table 10-11 describes the PIR fields.

Table 10-11. PIR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	P0	Processor 0 core reset. Setting this bit causes the PIC unit to assert the *core_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 10-7. There is one IPI vector/priority register per channel. The VECTOR and PRIORITY values should not be changed while IPIVPR*n*[A] is set.

	0	1	2								11	12	15	16	31
R	MSK	A	0	0	0	0	0	0	0	0	0	PRIORITY		VECTOR	
W															
Reset	1000_0000_0000_0000_0000_0000_0000_0000														
Offset	0x4_10A0, 0x4_10B0, 0x4_10C0, 0x4_10D0														

Figure 10-7. IPI Vector/Priority Registers (IPIVPR*n*)

Table 10-12 describes the IPIVPR*n* fields.

Table 10-12. 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 reported or is in-service. Note that this field is read only. The VECTOR and PRIORITY values should not be changed while IPIVPR*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
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 (IRR) shown in Figure 10-37.

10.3.1.6 Spurious Vector Register (SVR)

The spurious vector register (SVR) shown in Figure 10-8 contains the 16-bit vector returned to the processor when the IACK register is read for a spurious interrupt. See Section 10.4.4, “Spurious Vector Generation,” for more information about the events and conditions that may cause a spurious vector fetch.

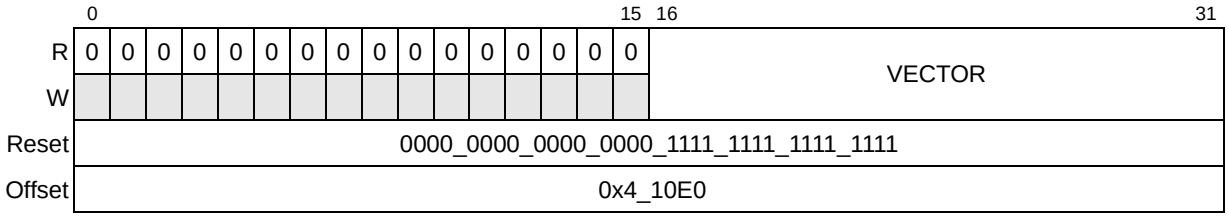


Figure 10-8. Spurious Vector Register (SVR)

Table 10-13 describes the SVR fields.

Table 10-13. SVR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	VECTOR	Spurious interrupt vector. Value returned when IACK is read during 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), shown in Figure 10-9, is written by software to report the clocking frequency of the PIC timers. Note that although TFRR is read/write, the value of this register is ignored by the PIC unit.

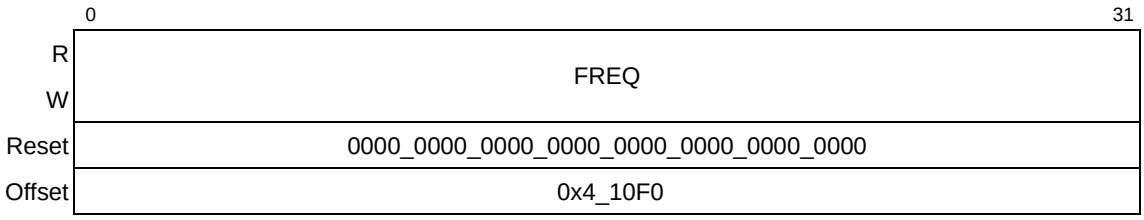


Figure 10-9. Timer Frequency Reporting Register (TFRR)

Table 10-14 describes the TFRR register.

Table 10-14. 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 (the core complex bus (CCB) clock) to user software. See Section 4.4.4, "Clocking," for more details. TFRR 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)."

10.3.2.2 Global Timer Current Count Registers (GTCCR*n*)

The global timer current count registers (GTCCRs), shown in Figure 10-10, contain the current count for each of the four PIC timers.

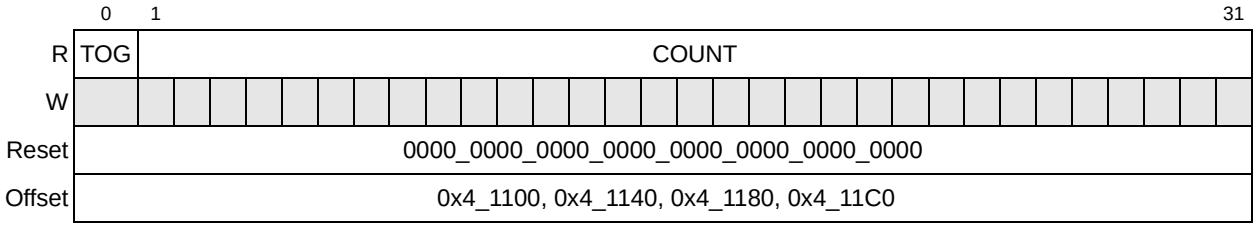


Figure 10-10. Global Timer Current Count Registers (GTCCR*n*)

Table 10-15 describes the GTCCR*n* fields.

Table 10-15. GTCCR*n* Field Descriptions

Bits	Name	Description
0	TOG	Toggle. Toggles when the current count decrements to zero. Cleared when GTBCR*n*[CI] goes from 1 to 0.
1–31	COUNT	Current count. Decremented while GTBCR*n*[CI] is zero. When the timer count reaches zero, an interrupt is generated (provided it is not masked), the toggle bit is inverted, and the count is reloaded. For non-cascaded timers, the reload value is the contents of the corresponding base count register. Cascaded timers are reloaded with either all ones, or the contents of the base count register, depending on the value of TCR[ROVR]. 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 (GTBCR*n*) contain the base counts for each of the four PIC timers as shown in Figure 10-11. This value is reloaded into the corresponding GTCCR*n* when the current count reaches zero. Note that when zero is written to the base count field, (and GTCCR*n*[CI] = 0), the timer generates an interrupt on every timer cycle.

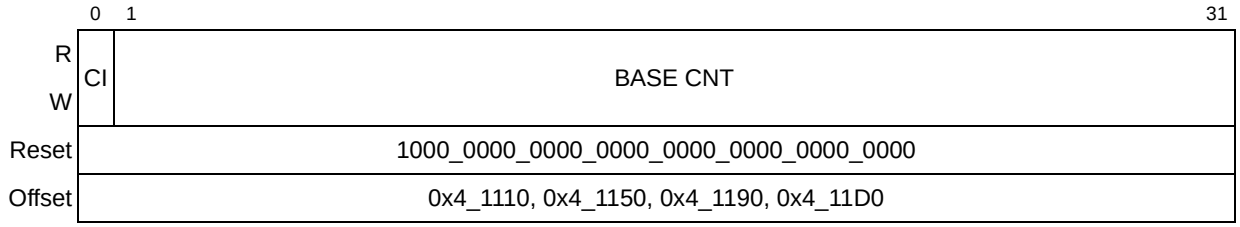


Figure 10-11. Global Timer Base Count Registers (GTBCR*n*)

Table 10-16 describes the GTBCR*n* fields.

Table 10-16. 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 CI transitions from 1 to 0, this value is copied into the corresponding current count register and the toggle bit is cleared. If 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 (GTVPR*n*) contain the interrupt vector and the interrupt priority as shown in Figure 10-12. They also contain the mask and activity fields.

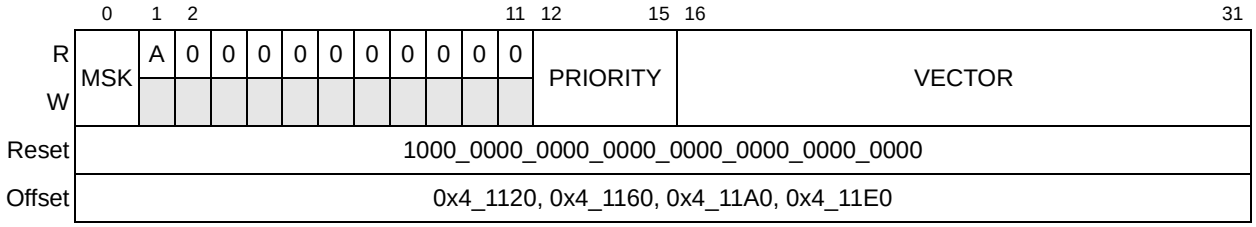


Figure 10-12. Global Timer Vector/Priority Registers (GTVPR*n*)

Table 10-17 describes the GTVPR*n* fields.

Table 10-17. 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.

Table 10-17. GTVPR*n* Field Descriptions (continued)

Bits	Name	Description
2–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 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 10-37.

10.3.2.5 Global Timer Destination Registers (GTDR*n*)

The global timer destination register controls the destination processor for this timer's interrupt, as shown in Figure 10-13.

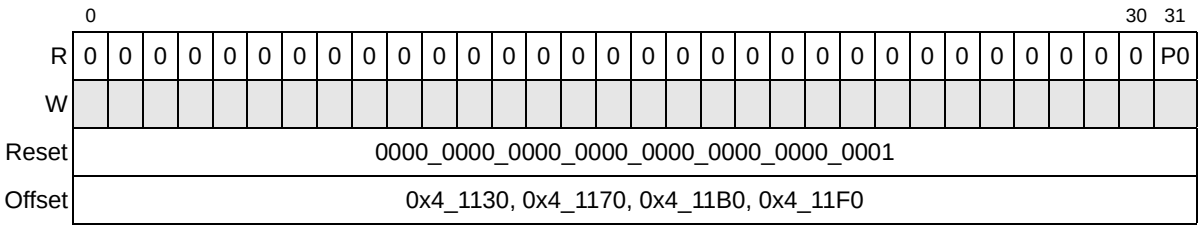


Figure 10-13. Global Timer Destination Registers (GTDR*n*)

Table 10-18 describes the GTDR*n* fields.

Table 10-18. GTDR*n* Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	P0	Processor 0. Indicates that processor 0 handles any interrupt. This bit is meaningful only in a multi-core device. Because the MPC8560 is a single-core device, internally serviced interrupts are always directed to processor 0. Permanently set and read only. 1 Interrupt directed to processor 0.

10.3.2.6 Timer Control Register (TCR)

The timer control register (TCR) shown in Figure 10-15 provides various configuration options such as count frequency and roll-over behavior.

There are two choices for the clock source for the timers: a selectable frequency ratio from the CCB clock, or the RTC signal. 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 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 10-19, given a CCB 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, generates one interrupt every hour from timer 2.

Table 10-19. Parameters for Hourly Interrupt Timer Cascade Example

System Clock	Clock Ratio	Timer Clock	Timer 0 Count	Timer 1 Count	Timer 2 Count
333 MHz	1 / 8	41.625 MHz	41.625×10^6 (0x027B_25A8)	59[1] (0x0000_0036)	59 (0x0000_0036)

[1] Counting down from 59 through 0 requires 60 ticks.

(41.625 x 106 ticks/sec) * (60 sec/min) * (60 min/hr) = total ticks/hr generating 1 interrupt/hr

Figure 10-14. Example Calculation for Cascaded Timers

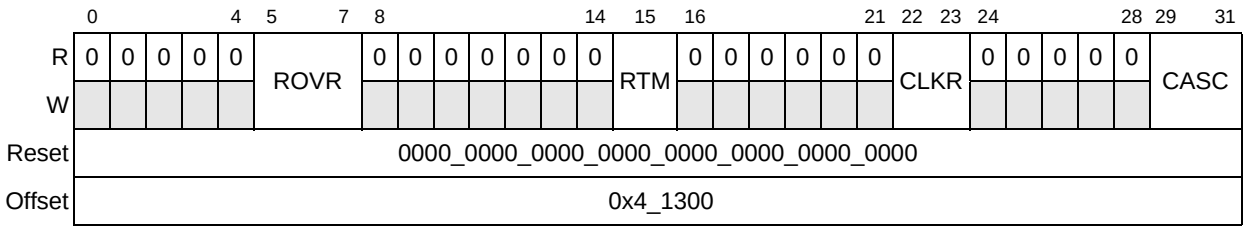


Figure 10-15. Timer Control Register (TCR)

Table 10-20 describes the TCR fields.

Table 10-20. TCR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	ROVR	Roll-over control for cascaded timers only. Specifies behavior when count reaches zero by identifying the source of the reload value. Cascaded timers are always reloaded with their base count value when the more significant timer in the cascade (the upstream timer) is zero. Bits 5–7 correspond to timers 2–0. Note that global timer 3 always reloads with its 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).
8–14	—	Reserved
15	RTM	Real time mode. Specifies the clock source for the PIC timers. 0 Timer clock frequency is a ratio of the frequency of the platform (CCB) clock as determined by the CLKR field. This is the default value. 1 The RTC signal is used to clock the PIC timers. If this bit is set, the CLKR field has no meaning.
16–21	—	Reserved
22–23	CLKR	Clock ratio. Specifies the ratio of the timer frequency to the platform (CCB) clock. The following clock ratios are supported: 00 Default. Divide by 8 01 Divide by 16 10 Divide by 32 11 Divide by 64
24–28	—	Reserved
29–31	CASC	Cascade timers. Specifies the output of particular global timers as input to others. 000 Default. Timers not cascaded 001 Cascade timers 0 and 1 010 Cascade timers 1 and 2 011 Cascade timers 0, 1, and 2 100 Cascade timers 2 and 3 101 Cascade timers 0 and 1; timers 2 and 3 110 Cascade timers 1, 2, and 3 111 Cascade timers 0, 1, 2, and 3

10.3.3 IRQ_OUT and Critical Interrupt Summary Registers

The summary registers indicate the interrupt sources directed to IRQ_OUT or *cint*. Summary register bits are cleared when the corresponding interrupt that caused a bit to be setþ is negated. Note that only level-sensitive interrupts can be directed to IRQ_OUT or *cint*.

10.3.3.1 IRQ_OUT Summary Register 0 (IRQSR0)

The IRQ_OUT summary registers contain one bit for each interrupt source. The corresponding bit is set if the interrupt is active and is directed to IRQ_OUT (that is, if the corresponding *x*IDR[EP] is set). Figure 10-16 shows IRQSR0.

	0															15	16	17	18	19	20										31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MSG0	MSG1	MSG2	MSG3	EXT*n*										
W																															
Reset	0000_0000_0000_0000_0000_0000_0000_0000																														
Offset	0x4_1310																														

Figure 10-16. IRQ_OUT Summary Register 0 (IRQSR0)

Table 10-21 describes the IRQSR0 fields.

Table 10-21. IRQSR0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	MSG0	Message interrupt 0 status 0 Interrupt is not active or not directed to IRQ_OUT. 1 Interrupt is active and is directed to the IRQ_OUT pin (that is, if the corresponding *x*IDR[EP] is set.
17	MSG1	Message interrupt 1 status 0 Interrupt is not active or not directed to IRQ_OUT. 1 Interrupt is active and is directed to the IRQ_OUT pin (that is, if the corresponding *x*IDR[EP] is set.
18	MSG2	Message interrupt 2 status 0 Interrupt is not active or not directed to IRQ_OUT. 1 Interrupt is active and is directed to IRQ_OUT (that is, if the EP corresponding *x*IDR[EP] is set.
19	MSG3	Message interrupt 3 status 0 Interrupt is not active or not directed to IRQ_OUT. 1 Interrupt is active and is directed to IRQ_OUT (that is, if corresponding *x*IDR[EP] is set.
20–31	EXT*n*	External interrupts 0–11. Each bit corresponds to a different interrupt according to the following: Bit Interrupt 20 IRQ0 21 IRQ1 . . . . 31 IRQ11 0 The corresponding interrupt is not active or not directed to IRQ_OUT. 1 The corresponding interrupt is active and directed to IRQ_OUT (if the corresponding *x*IDR[EP] is set.

10.3.3.2 $\overline{\text{IRQ\_OUT}}$ Summary Register 1 (IRQSR1)

Figure 10-17 shows IRQSR1.

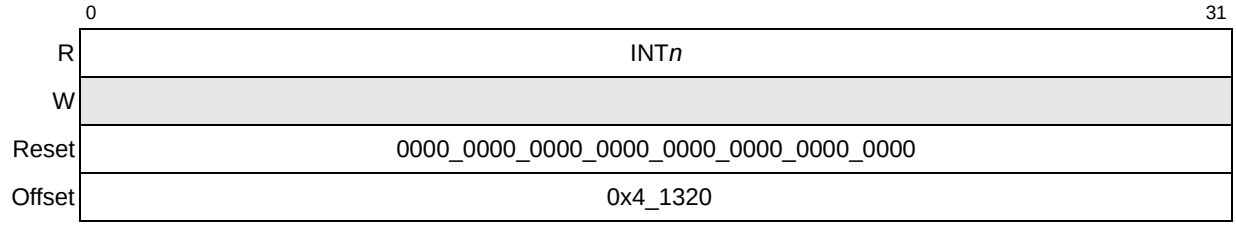


Figure 10-17. $\overline{\text{IRQ\_OUT}}$ Summary Register 1 (IRQSR1)

Table 10-22 describes the IRQSR1 fields.

Table 10-22. IRQSR1 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 0–31 status. Bit 0 represents INT0. Bit 31 represents INT31. 0 The corresponding interrupt is not active or not directed to $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and is directed to $\overline{\text{IRQ\_OUT}}$ (that is, if the corresponding *x*IDR[EP] is set).

10.3.3.3 Critical Interrupt Summary Register 0 (CISR0)

The critical interrupt summary registers 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* (if the CI field in its corresponding destination register is set). Figure 10-18 shows CISR0.

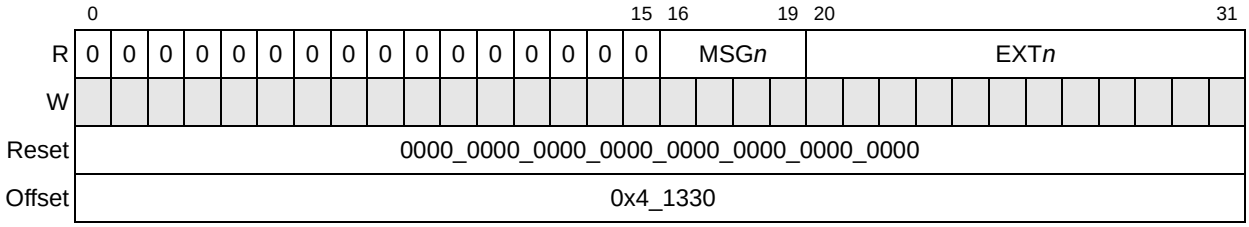


Figure 10-18. Critical Interrupt Summary Register 0 (CISR0)

Table 10-23 describes the CISR0 fields.

Table 10-23. CISR0 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–19	MSG*n*	Message interrupts 0–3. Bit 16 represents MSG0; bit 19 represents MSG3. 0 The corresponding interrupt is not active or not directed to *cint*. 1 The corresponding interrupt is active and is directed to the *cint* (if the corresponding xIDR[CI] is set).
20–31	EXT*n*	External interrupts 0–11. Bit 20 represents IRQ0. Bit 31 represents IRQ11. 0 The corresponding interrupt is not active or not directed to *cint*. 1 The corresponding interrupt is active and is directed to the *cint* (if the corresponding xIDR[CI] is set).

10.3.3.4 Critical Interrupt Summary Register 1 (CISR1)

Figure 10-19 shows CISR1.

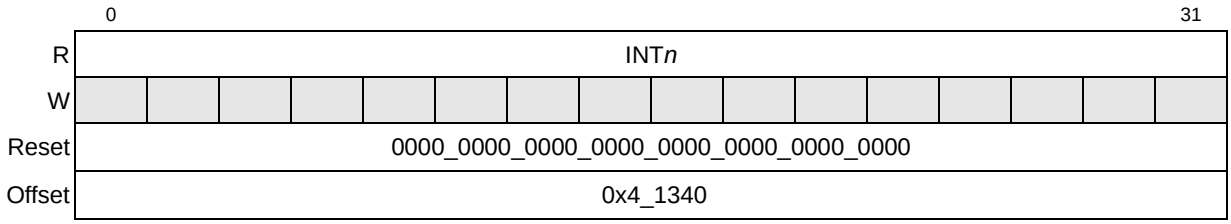


Figure 10-19. Critical Interrupt Summary Register 1 (CISR1)

Table 10-24 describes CISR1 register.

Table 10-24. CISR1 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 0–31. Bit 0 represents INT0. Bit 31 represents INT31. 0 Corresponding interrupt is not active or not directed to *cint*. 1 The corresponding interrupt is active and is directed to the *cint* (if the corresponding xIDR[CI] is set).

10.3.4 Performance Monitor Mask Registers (PMMRs)

There are four pairs of performance monitor mask registers, PM*n*MR0 and PM*n*MR1. Each pair can be configured to select one interrupt source (interprocessor, timer, message, external, or internal) to generate a performance monitor event. The performance monitor can be configured to track this event in the performance monitor local control registers. See Section 18.3.2.2, “Performance Monitor Local Control Registers (PMLCAn and PMLCBn).”

10.3.4.1 Performance Monitor Mask Register (Lower) (PM*n*MR0)

Figure 10-20 shows the PM*n*MR0 registers. Each register is paired with a PM*n*MR1 register. Because each unreserved bit in the 64-bit pair (PM*n*MR0/1) specifies a different interrupt, only one bit in each pair can be unmasked at a time. Unmasking more than one bit per pair is considered a programming error and results in unpredictable behavior.

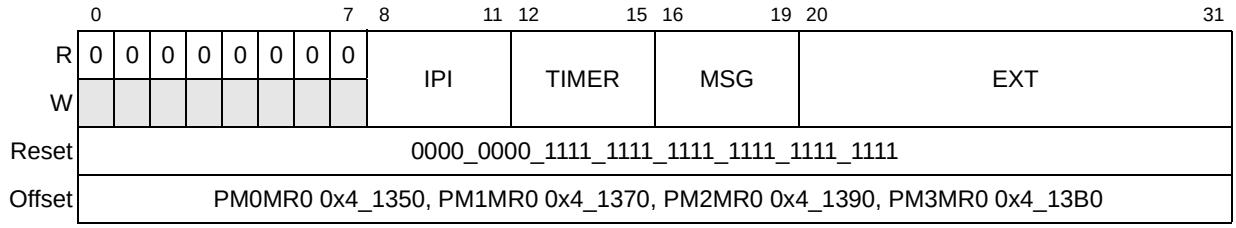


Figure 10-20. Performance Monitor Mask Registers (PM*n*MR0)

Table 10-25 describes the PM*n*MR0 fields.

Table 10-25. PM*n*MR0 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	IPI	IPI interrupts 0–3 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
12–15	TIMER	Timer interrupts 0–3 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
16–19	MSG	Message interrupts 0–3 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
20–31	EXT	External interrupts IRQ[0:11] 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

10.3.4.2 Performance Monitor Mask Registers (Upper) (PM*n*MR1)

Figure 10-21 shows the PM0MR1–PM3MR1 fields.

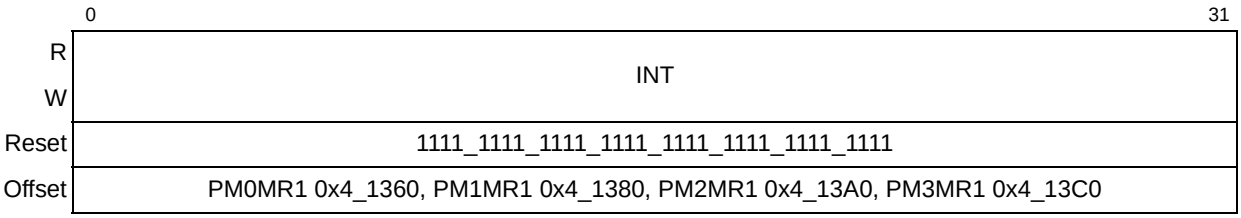


Figure 10-21. Performance Monitor Mask Registers (PM*n*MR1)

Table 10-26 describes the PM*n*MR1 registers.

Table 10-26. PM*n*MR1 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 0–31 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

10.3.5 Message Registers

Writing to one of the four message registers (MSGR0–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 PIC message status register (MSR), shown in Figure 10-24.

10.3.5.1 Message Registers (MSGR0–MSGR3)

The message registers (MSGR0–MSGR3) are shown in Table 10-22.

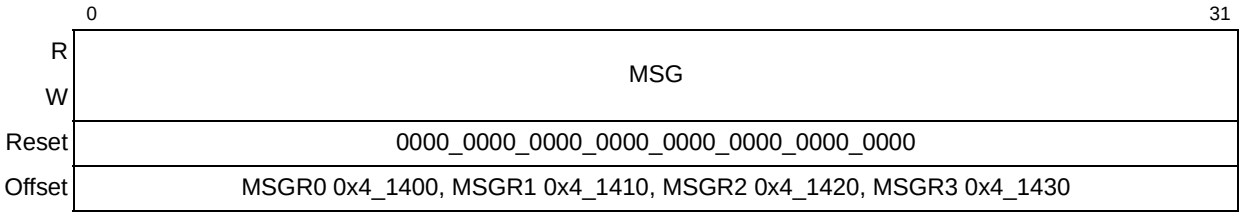


Figure 10-22. Message Registers (MSGRs)

Table 10-27 describes the MSGR registers.

Table 10-27. 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 10-23 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.

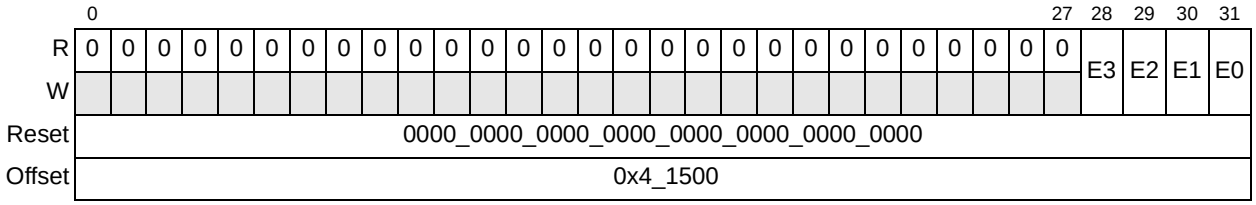


Figure 10-23. Message Enable Register (MER)

Table 10-28 describes the MER fields.

Table 10-28. MER Field Descriptions

Bits	Name	Description
0–27	—	Reserved
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 PIC message status register (MSR) shown in Figure 10-24 contains status bits for each message register. The status bit is set when the corresponding messaging interrupt is active. Reading the corresponding message register or writing a 1 to a status bit clears the corresponding message interrupt and the status bit.

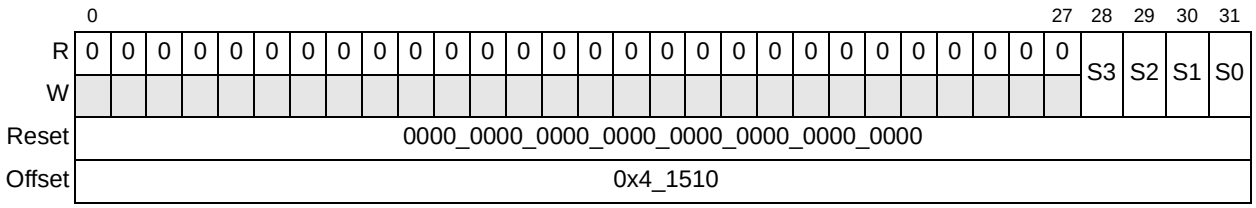


Figure 10-24. Message Status Register (MSR)

Table 10-29 describes the MSR fields.

Table 10-29. MSR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
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.
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.

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]. The format of the EIVPRs is shown in Figure 10-25.

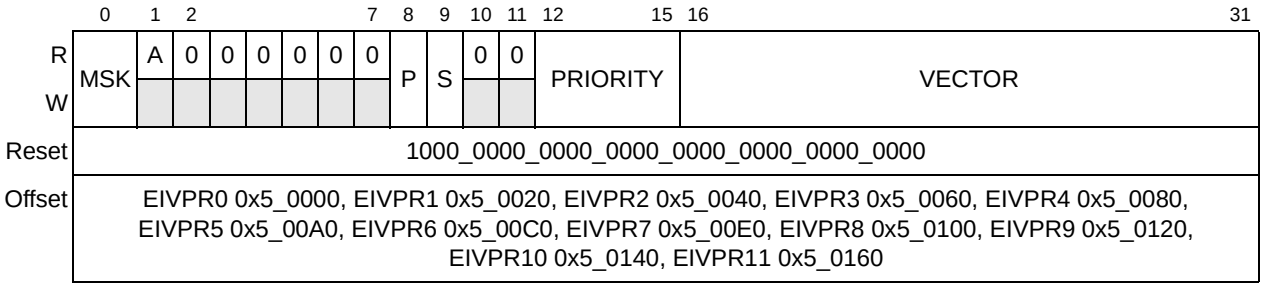


Figure 10-25. External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)

Table 10-30 describes the EIVPR fields.

Table 10-30. 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.
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.
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 (IRR) shown in Figure 10-37.

10.3.6.2 External Interrupt Destination Registers (EIDR0–EIDR11)

The external interrupt destination registers (EIDR*n*), shown in Figure 10-26, 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. 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 EIDR*n* fields.

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																											30	31
R	EP	CI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P0
W																															
Reset	0000_0000_0000_0000_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 10-26. External Interrupt Destination Registers (EIDR*n*)

Table 10-31 describes the EIDR fields. Because external interrupts can be channeled only to processor 0, the P0 bit is permanently set. As shown in Figure 10-37, if either the CI or EP bits 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.

Table 10-31. 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
31	P0	Processor 0. Indicates that processor 0 handles any interrupt. This bit is meaningful only 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. Permanently set and read only. 1 Interrupt directed to processor 0.

10.3.6.3 Internal Interrupt Vector/Priority Registers (IIVPR0–IIVPR31)

The internal interrupt vector/priority registers (IIVPR*n*), shown in Figure 10-27, have the same fields and format as the GTVPR*n*, except that they apply to the internal interrupt sources listed in Table 10-4. These interrupts are all level-sensitive.

NOTE

Because all internal interrupts are active-high, clearing any IIVPR*n* polarity field disables that interrupt. Care should be taken to ensure this field is not inadvertently corrupted when loading or reloading IIVPRs with priority, mask, or vector data.

	0	1	2					7	8	9			11	12	15	16	31
R	MSK	A	0	0	0	0	0	0	P	0	0	0		PRIORITY		VECTOR	
W																	
Reset	1000_0000_1000_0000_0000_0000_0000_0000																
Offset	IIVPR0–7: 0x5_0200, 0x5_0220, 0x5_0240, 0x5_0260, 0x5_0280, 0x5_02A0, 0x5_02C0, 0x5_02E0 IIVPR8–15: 0x5_0300, 0x5_0320, 0x5_0340, 0x5_0360, 0x5_0380, 0x5_03A0, 0x5_03C0, 0x5_03E0 IIVPR16–23: 0x5_0400, 0x5_0420, 0x5_0440, 0x5_0460, 0x5_0480, 0x5_04A0, 0x5_04C0, 0x5_04E0 IIVPR24–31: 0x5_0500, 0x5_0520, 0x5_0540, 0x5_0560, 0x5_0580, 0x5_05A0, 0x5_05C0, 0x5_05E0																

Figure 10-27. Internal Interrupt Vector/Priority Registers (IIVPR*n*)

Table 10-32 describes the IIVPR*n* fields.

Table 10-32. 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
8	P	Polarity. Specifies the polarity for the internal interrupt. Note: Because all internal interrupts are active-high, clearing this bit disables the interrupt. 0 Interrupt polarity is active-low. This value disables the interrupt. 1 Interrupt polarity is active-high. This is the reset value should not be changed.
9–11	—	Reserved.
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 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 (IRR) shown in Figure 10-37.

10.3.6.4 Internal Interrupt Destination Registers (IIDR0–IIDR31)

Internal interrupt destination registers (IIDR*n*), shown in Figure 10-28, have the same fields and format as EIVDR*n*, except that they apply to the internal interrupt sources listed in Table 10-4. Like external interrupts, internal interrupts can be directed only to processor 0.

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																												30	31
R	EP	CI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P0
W																																
Reset	0000_0000_0000_0000_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 10-28. Internal Interrupt Destination Registers (IIDR*n*)

Table 10-33 describes the IIDR*n* fields.

Table 10-33. 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. The behavior of the PIC is not defined if both EP and CI are set on the same interrupt destination register.
1	CI	Critical interrupt. 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. The behavior of the PIC is not defined if both EP and CI are set on the same interrupt destination register.
2–30	—	Reserved
31	P0	Processor 0. Indicates that processor 0 handles any interrupt. This bit is meaningful only 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. 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 (MIVPR*n*), shown in Figure 10-29, have the same fields and format as the GTVPR*n*, except they apply to messaging interrupts.

	0	1	2								11	12	15	16	31
R	MSK	A	0	0	0	0	0	0	0	0	0	PRIORITY		VECTOR	
W															
Reset	1000_0000_0000_0000_0000_0000_0000_0000														
Offset	0x5_1600, 0x5_1620, 0x5_1640, 0x5_1660,														

Figure 10-29. Messaging Interrupt Vector/Priority Registers (MIVPR*n*)

Table 10-34 describes the MIVPR*n* fields.

Table 10-34. 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

Table 10-34. MIVPR*n* Field Descriptions (continued)

Bits	Name	Description
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 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 (IRR) shown in Figure 10-37.

10.3.6.6 Messaging Interrupt Destination Registers (MIDR0–MIDR3)

The messaging interrupt destination registers (MIDR*n*), shown in Figure 10-30, control the destination for the messaging interrupts. 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*).

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																												30	31
R	EP	CI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	P0
W																																
Reset	0000_0000_0000_0000_0000_0000_0000_0001																															
Offset	0x5_1610, 0x5_1630, 0x5_1650, 0x5_1670																															

Figure 10-30. Messaging Interrupt Destination Registers (MIDR*n*)

Table 10-35 describes the MIDR*n* fields.

Table 10-35. 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. The behavior of the PIC is not defined if both EP and CI are set on the same interrupt destination register.
1	CI	Critical interrupt 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. The behavior of the PIC is not defined if both EP and CI are set on the same interrupt destination register.
2–30	—	Reserved
31	P0	Processor 0. Indicates that processor 0 handles any interrupt. This bit is meaningful only 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. Permanently set and read only. 1 Interrupt directed to processor 0.

10.3.7 Per-CPU Registers

The OpenPIC programming model supports multiprocessor systems of up to 32 separate processors. As such, the OpenPIC interface specification provides for coordinating both the requesting and servicing of interrupts among several processor cores within a single integrated device. To fully comply with the OpenPIC specification, the 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 in Table 10-36 are called per-CPU registers, because they would be duplicated for each core in a multi-core device. The OpenPIC interface specifies that a copy of these registers be available to each core at the same physical address by using the ID of the processor that initiates the transaction to determine the set of per-CPU registers to access.

Table 10-36. Per-CPU Registers—Private Access Address Offsets

Register Name	Offset
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 (WHOAMI)	0x4_0090
Interrupt acknowledge register (IACK)	0x4_00A0
End of interrupt register (EOI)	0x4_00B0

These addresses, shown in Table 10-36, 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 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 multiprocessor 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 10-31. Note that when accessing a register normally, each core sources a different address. However, when accessing the same register using the per-CPU address space, each core sources the same address.

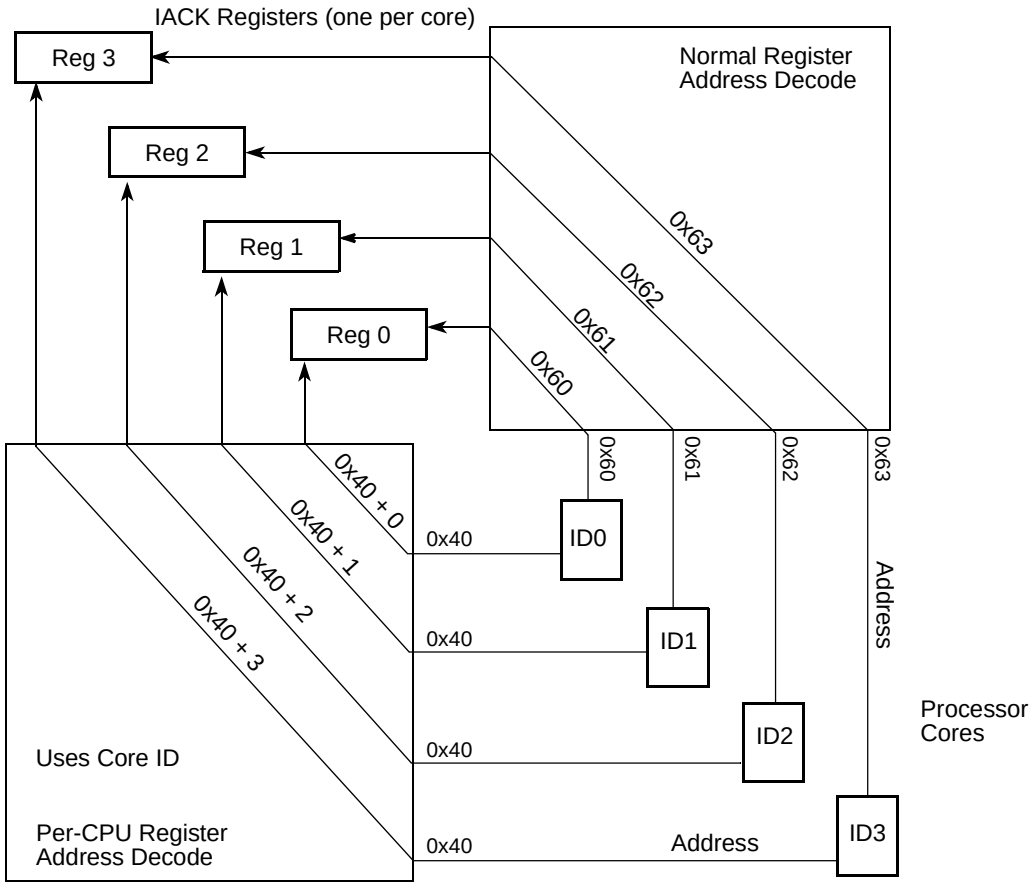


Figure 10-31. Per-CPU Register Address Decoding in a Four-Core Device

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 10-32. Writing to an IPIDR with a bit set causes a self interrupt. Interprocessor interrupts are of limited usefulness in a single-core device; however, this feature can serve as a doorbell type interrupt because external bus masters can write to these registers.

	0																												30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																														P0
Reset	0000_0000_0000_0000_0000_0000_0000_0000																													
Offset	IPIDR0 0x6_0040, IPIDR1 0x6_0050 IPIDR2 0x6_0060, IPIDR3 0x6_0070 (see Section 10.3.7, “Per-CPU Registers.”)																													

Figure 10-32. Interprocessor Interrupt Dispatch Registers (IPIDR0–IPIDR3)

Table 10-37 describes the IPIDR*n* fields.

Table 10-37. IPIDR*n* Field Descriptions

Bits	Name	Description
0–30	—	Reserved
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 (MPC8560) shown in Figure 10-33. Software must write the priority of the current processor task in the CTPR. The PIC uses this value for comparison with the priority of incoming interrupts. Given several concurrent incoming interrupts, the highest priority interrupt is asserted to the core if the following apply:

- The interrupt is not masked
- The priority of the interrupt 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 external interrupts signaled through the *int* signal. Hardware sets CTPR to 0x0000_000F during reset or when the Px field of the processor initialization register is set. This effectively prevents the PIC from signaling any interrupts through the *int* signal.

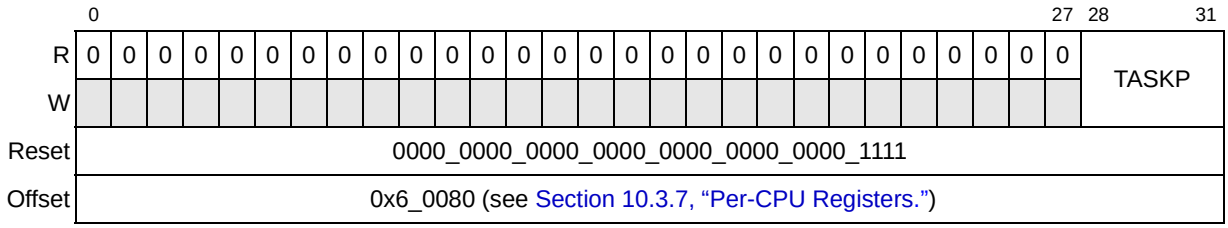


Figure 10-33. Processor Current Task Priority Register (CTPR)

Table 10-38 describes the CTPR.

Table 10-38. CTPR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	TASKP	Task priority. This field is set from 0 to 15, where 15 corresponds to the highest priority for processor tasks. If 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), shown in Figure 10-34, 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.

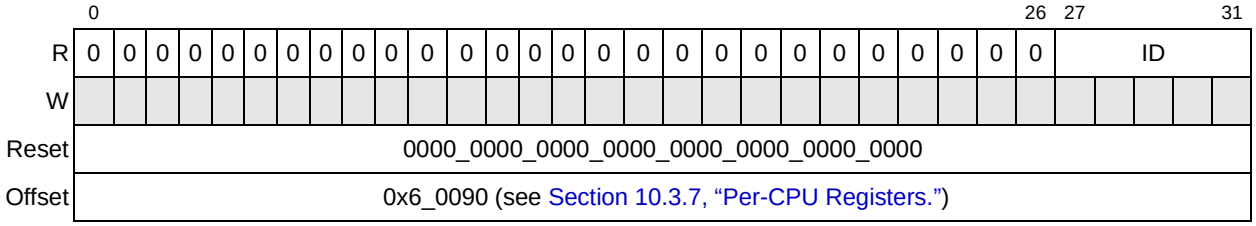


Figure 10-34. Processor Who Am I Register (WHOAMI)

Table 10-39 describes the WHOAMI fields.

Table 10-39. WHOAMI Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27–31	ID	ID. Returns the ID of the processor reading this register. Always returns zero. 00000 Processor core

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 operation to a memory-mapped interrupt acknowledge register (IACK), shown in Figure 10-35. 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 (IPR) is cleared for edge-sensitive interrupts to indicated that the interrupt is no longer pending.
- The corresponding field in the in-service register (ISR) is set to indicated that the interrupt has been signaled to the core.
- The interrupt signal (*int*) to the processor is negated.

Reading this register when there is no pending interrupt returns the spurious vector value as described in Section 10.3.1.6, “Spurious Vector Register (SVR).”

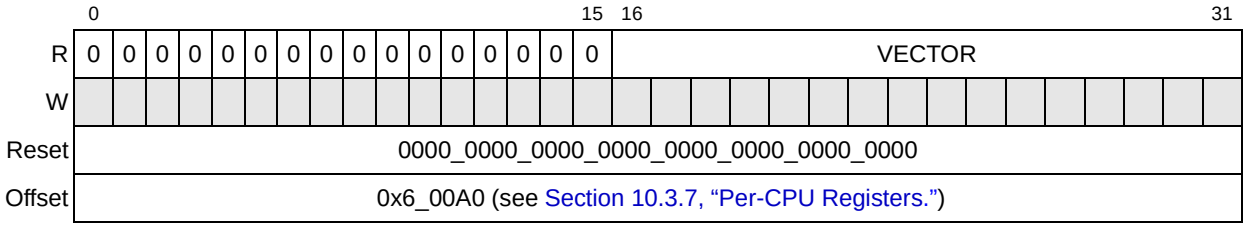


Figure 10-35. Processor Interrupt Acknowledge Register (IACK)

Table 10-40 describes the IACK fields.

Table 10-40. IACK Field Descriptions

Bits	Name	Description
0–15	—	Reserved
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 10-36 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.

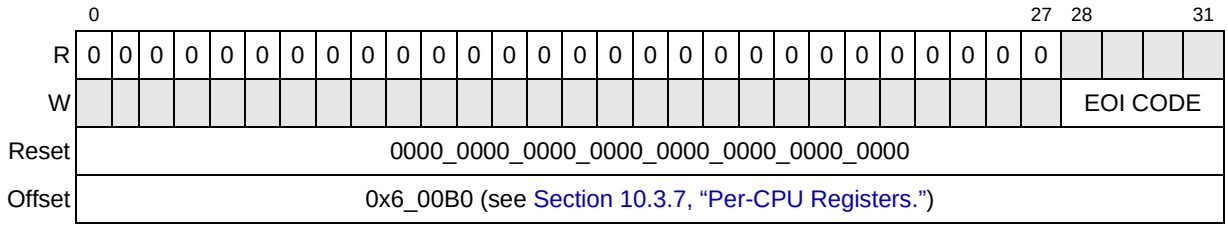


Figure 10-36. End of Interrupt Register (EOI)

Table 10-41 describes the EOI fields.

Table 10-41. EOI Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EOI CODE	0000 (write only)

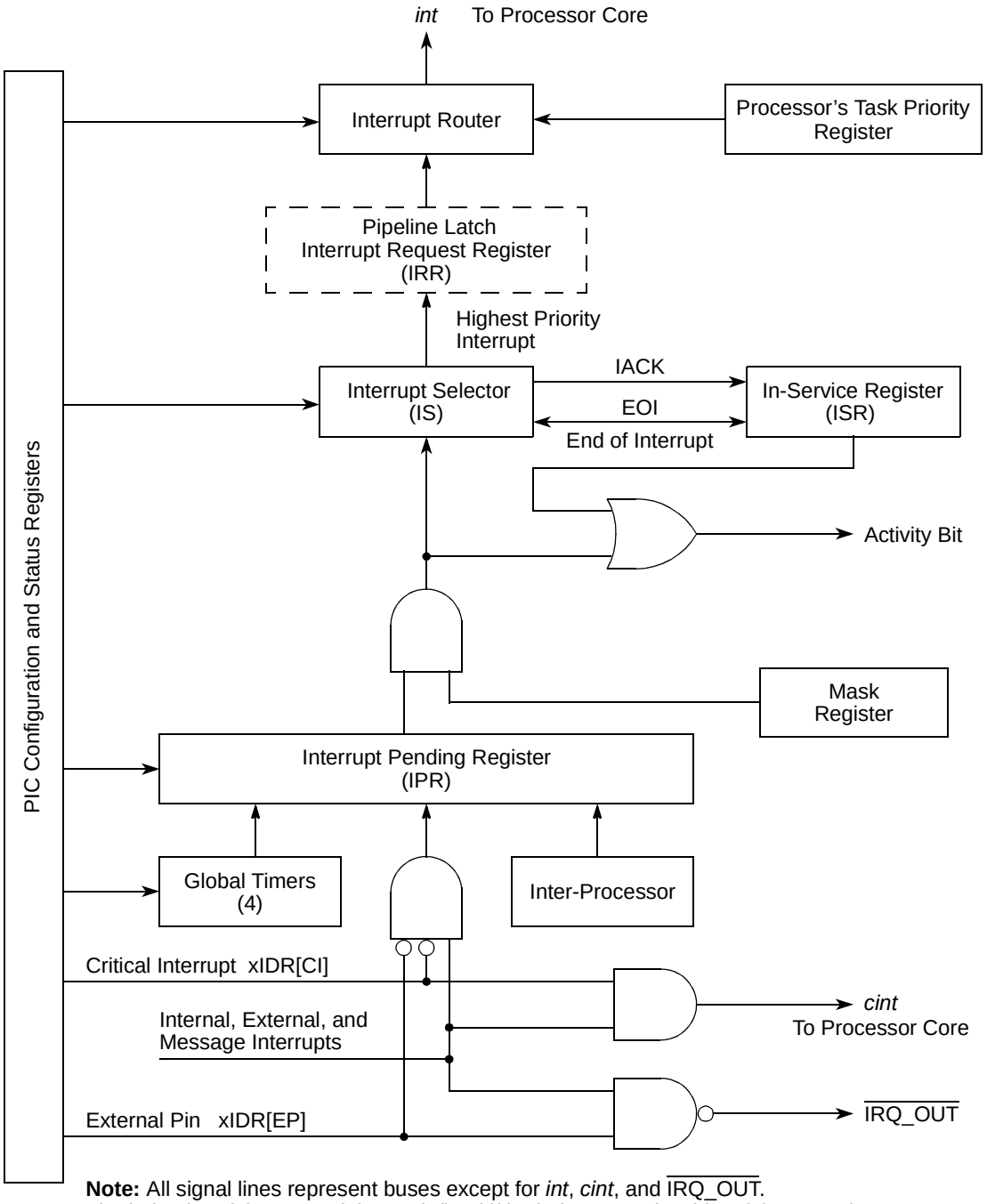
10.4 Functional Description

This section is a functional description of the PIC.

10.4.1 Flow of Interrupt Control

Figure 10-37 is a block diagram of the PIC unit showing the flow of interrupt processing. This figure is intended to aid in understanding and does not fully represent all internal circuitry of the actual implementation. The PIC receives interrupt signals from both external and internal sources. These signals are qualified and latched in the interrupt pending register (IPR). The IPR feeds the interrupt selector (IS). The interrupt router of the PIC monitors the outputs of its internal interrupt request register (IRR) and other configuration registers. When the priority of the interrupt latched in the IRR is higher than the value in the processor's task priority register, the interrupt router 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.

Note that the IPR, IS, and IRR are internal registers that are not accessible to the programmer.



Note: All signal lines represent buses except for *int*, *cint*, and $\overline{\text{IRQ\_OUT}}$.
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 10-37. PIC Interrupt Processing Flow Diagram

The interrupt handler executing on the processor should then acknowledge the interrupt by explicitly reading IACK (at which point the interrupt is considered to be in-service). The PIC unit interprets this read as an interrupt acknowledge (IACK) cycle; in response, the PIC unit returns the

vector associated with the interrupt source to the interrupt handler routine. The handler can further vector to different branches of interrupt handling accordingly.

Note that reading IACK also negates the interrupt signal to the processor. See Section 10.3.7.4, "Processor Interrupt Acknowledge Register (IACK)," for more details.

The internal in-service register (ISR) tracks all interrupts that have caused *int* to be asserted and that have received an IACK cycle. 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.

10.4.1.1 Interrupt Source Priority

Each interrupt source is assigned a priority value of 0–15 through its corresponding vector/priority register, where 15 is the highest priority. Interrupts are delivered only when the priority in the vector/priority is greater than the priority programmed into the current task priority register, CTPR. Setting a source priority to zero inhibits that interrupt. Likewise, setting the CTPR value to 15 disables all interrupts directed to the *int* signal.

The PIC unit services simultaneous interrupts occurring with the same priority according to the following order:

1. MSG0–MSG3
2. IPI0–IPI3
3. Timer0–timer3
4. IRQ[0:11]
5. Internal0–internal31

For example, if MSG0, MSG2, and IPI0 have the same priority and generate simultaneous interrupts, they are serviced in the following order:

1. MSG0
2. MSG2
3. IPI0

10.4.1.2 Processor Current Task Priority

The CTPR 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.

10.4.1.3 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.

10.4.2 Nesting of Interrupts

If the processor is servicing an interrupt, it can only be interrupted again if the PIC receives an interrupt request from a source with higher priority than the one currently being serviced. This is true even if software, as part of its interrupt service routine, writes a new and lower value into the CTPR.

Thus, although several interrupts may be in-service simultaneously, the code currently executing is always handling the highest priority of all the interrupts that are in service. 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. An interrupt with lower priority than those currently in-service is not started until all higher priority interrupts complete even if its priority is greater than the CTPR value.

10.4.3 Processor Initialization

The processor can reset itself by writing to the processor initialization register (PIR). This causes the assertion of the *core_reset* output signal. When this occurs, the processor also gets written to 0x000F to disable the delivery of any interrupts.

10.4.4 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 that is activated with level-sensitive logic and the source negates before the interrupt is acknowledged.
- *int* is asserted for an interrupt source that is later masked (using the mask bit in the corresponding vector/priority register) 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 PIC did not assert the *int* signal.

In all cases, a spurious vector is not returned if another pending interrupt has sufficient priority to interrupt the processor. If such an interrupt is available, the vector for that 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.

10.4.5 Messaging Interrupts

There are four 32-bit message registers that can be used to send 32-bit messages to the processor. A messaging interrupt is generated by writing 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.

10.4.6 Global 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 the following four configuration and control registers:

- Global timer current count register (GTCCR*n*)
- Global timer base count register (GTBCR*n*)
- Global timer vector-priority register (GTVPR*n*)
- Global timer destination register (GTDR*n*)

The timer frequency should be written to the TFRR. (All of the timers operate 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 10.3.2.6, “Timer Control Register (TCR).”

10.4.7 Reset of the PIC

The PIC unit is reset by a device power-on reset (POR) or by software that sets the GCR[RST] bit. Both of these actions cause 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.
- PIR, TFRR, TCR, MER, MSR, and MSGR0–3 are cleared.
- MSG and timer destination fields are set.
- The IPI dispatch registers are 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 interrupt vector resets to 0xFFFF.
- The PMMRs are reset to 0xFFFF.
- The PIC defaults to the pass-through mode (GCR[M] = 0).
- All other registers remain at their pre-reset programmed values.

The GCR[RST] bit is automatically cleared when the reset sequence is complete.

10.5 Initialization/Application Information

This section contains initialization and application information for the PIC.

10.5.1 Programming Guidelines

The following subsections contain information about programming PIC registers.

10.5.1.1 PIC Registers

Most PIC control and status registers are readable and return the last value written. The exceptions to this rule are as follows:

- IPI dispatch registers and the EOI register, 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] = 1):

- All PIC registers must be located in a cache-inhibited and guarded area (through the processor MMU).
- The PIC portion of the address map must be set-up appropriately.

In addition, the following initialization sequence is recommended:

1. Write the vector, priority, and polarity values in each interrupt's vector/priority register, leaving their MSK (mask) bit set. This is required only if interrupts are used.
2. Clear the CTPR (CTPR = 0x0000_0000).
3. Program the PIC to mixed mode by setting GCR[M].
4. Clear the MSK bit in the vector/priority registers to be used.
5. Perform a software loop to clear all pending interrupts:
   — Load counter with 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.

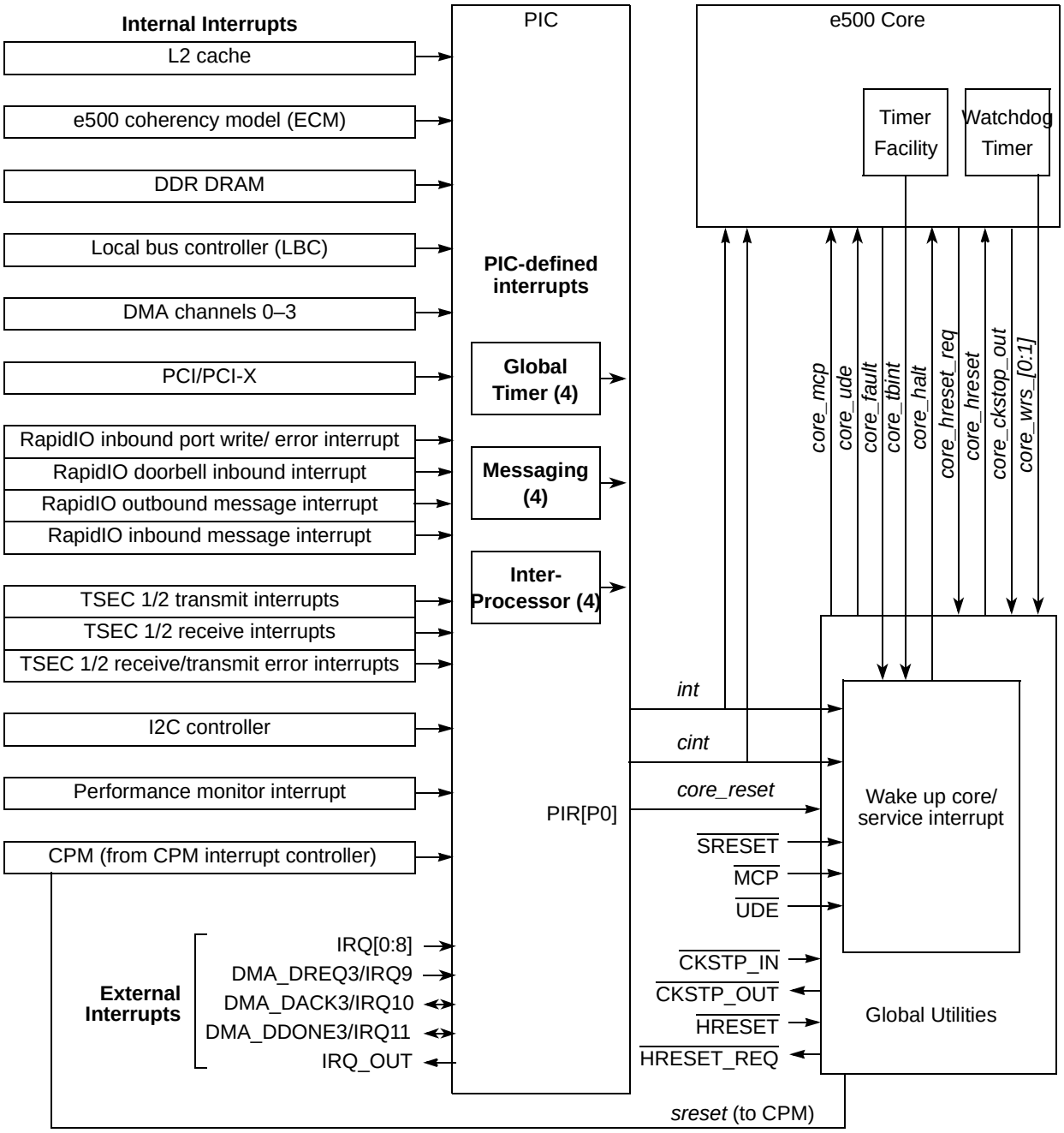
To avoid having to handle a false interrupt for this case, software can clear the PIC interrupt pending register of these spurious edge detections by first configuring the polarity/sense of external interrupt sources to be level-sensitive; high-level if the input is a positive-edge source, low-level if it's a negative-edge source (while the mask bit remains set). After this is complete, configuring the external interrupt source as edge-sensitive will not cause a false interrupt.

10.5.1.2 Changing Interrupt Source Configuration

To change the vector, priority, polarity, sense or destination of an active (unmasked) interrupt source, the following 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.



Chapter 11
I²C Interface

This chapter describes the inter-IC (IIC or I²C) bus interface implemented on this device.

11.1 Introduction

The I²C bus is a two-wire—serial data (SDA) and serial clock (SCL)— bidirectional serial bus that provides a simple efficient method of data exchange between this device and other devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. Figure 11-1 shows a block diagram of the I²C interface.

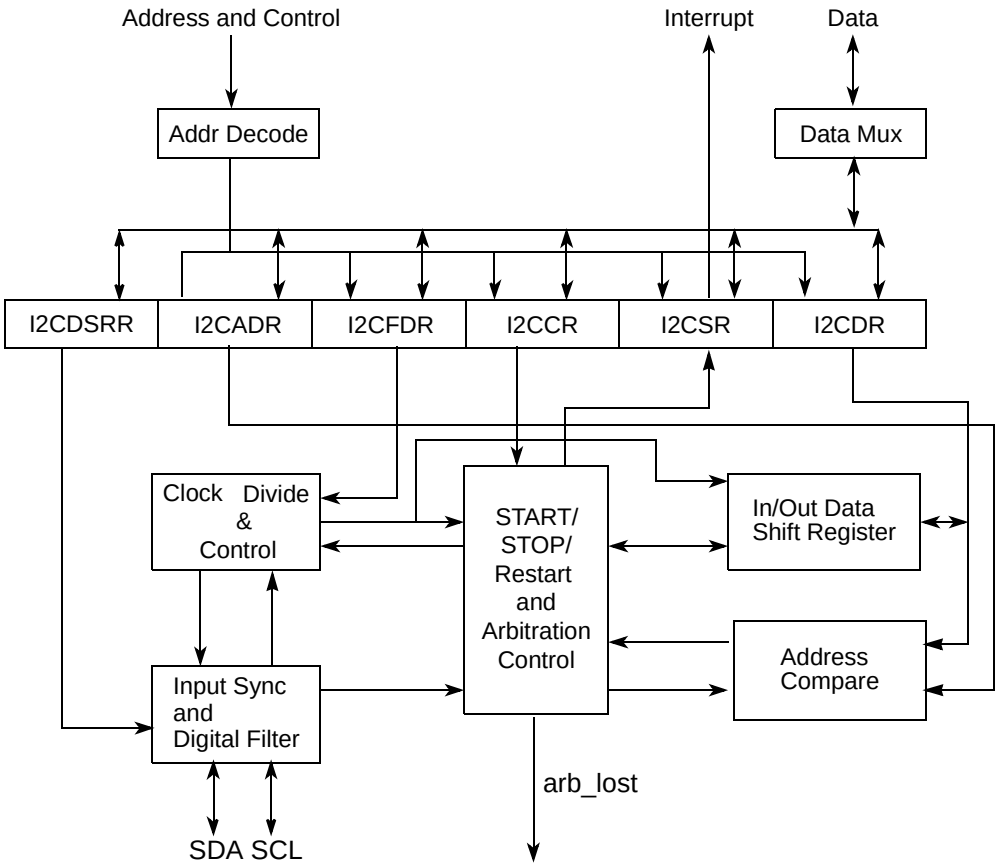


Figure 11-1. I²C Block Diagram

11.1.1 Overview

The two-wire I²C bus minimizes interconnections between devices. The synchronous, multiple-master I²C bus allows the connection of additional devices to the bus for expansion and system development. The bus includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously.

11.1.2 Features

The I²C interface includes the following features:

- Two-wire interface
- Multiple-master operation
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- START and STOP signal generation/detection
- Acknowledge bit generation/detection
- Bus busy detection
- Software-programmable clock frequency
- Software-selectable acknowledge bit
- On-chip filtering for spikes on the bus

11.1.3 Modes of Operation

The I²C unit on this device can operate in one of the following modes:

- Master mode—The I²C is the driver of the SDA line. It cannot use its own slave address as a calling address. The I²C cannot be a master and a slave simultaneously.
- Slave mode—The I²C is not the driver of the SDA line. The module must be enabled before a START condition from a non-I²C master is detected.
- Interrupt-driven byte-to-byte data transfer—When successful slave addressing is achieved (and SCL returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/$\overline{W}$ bit sent by the calling master. Each byte of data must be followed by an acknowledge bit, which is signalled from the receiving device. Several bytes can be transferred during a data transfer session.
- Boot sequencer mode—This mode can be used to initialize the configuration registers in the device after the I²C module is initialized. Note that the MPC8560 powers up with boot sequencer mode disabled as a default, but this mode can be selected with the cfg_boot_seq[0:1] power-on reset (POR) configuration signals that are located on the LGPL3 and LGPL5 signals.

Additionally, the following three I^2C-specific states are defined for the I^2C interface:

- START condition—This condition denotes the beginning of a new data transfer (each data transfer contains several bytes of data) and awakens all slaves.
- Repeated START condition—A START condition that is generated without a STOP condition to terminate the previous transfer.
- STOP condition—The master can terminate the transfer by generating a STOP condition to free the bus.

11.2 External Signal Descriptions

The following sections give an overview of signals and provide detailed signal descriptions.

11.2.1 Signal Overview

The I^2C interface uses the SDA and SCL signals, described in Table 11-1, for data transfer. All devices 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. 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

SDA and SCL, described in Table 11-2, serve as a communication interconnect with other devices. All devices connected to these two signals must have open-drain or open-collector outputs. The logic AND function is performed on both of these signals with external pull-up resistors. Refer to the *MPC8560 Integrated Communications Processor Hardware Specifications* for the electrical characteristics of these signals.

Table 11-2. I²C Interface Signal—Detailed Signal Descriptions

Signal	I/O	Description
SCL	I/O	Serial clock. Performs as an input when the MPC8560 is programmed as an I²C slave. SCL also performs as an output when the MPC8560 is programmed as an I²C master.
	O	As outputs for the bidirectional serial clock, these signals operate as described below.
		State Meaning Asserted/Negated—Driven along with SDA as the clock for the data.
	I	As inputs for the bidirectional serial clock, these signals operate as described below.
		State Meaning Asserted/Negated—The I²C unit uses this signal to synchronize incoming data on SDA. The bus is assumed to be busy when this signal is detected low.
SDA	I/O	Serial data. 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 I²C master or a slave).
	O	As outputs for the bidirectional serial data, these signals operate as described below.
		State Meaning Asserted/Negated— Data is driven.
	I	As inputs for the bidirectional serial data, these signals operate as described below.
		State Meaning Asserted/Negated—Used to receive data from other devices. The bus is assumed to be busy when SDA is detected low.

11.3 Memory Map/Register Definition

Table 11-3 lists the I²C–specific registers and their addresses.

Table 11-3. I²C Memory Map

Address	I²C Register	Access	Reset	Section/Page
0x0_3000	I2CADR—I²C address register	R/W	0x00	11.3.1.1/11-5
0x0_3004	I2CFDR—I²C frequency divider register	R/W	0x00	11.3.1.2/11-5
0x0_3008	I2CCR—I²C control register	R/W	0x00	11.3.1.3/11-6
0x0_300C	I2CSR—I²C status register	R/W	0x81	11.3.1.4/11-7
0x0_3010	I2CDR—I²C data register	R/W	0x00	11.3.1.5/11-9
0x0_3014	I2CDFSRR—I²C digital filter sampling rate register	R/W	0x10	11.3.1.6/11-10

11.3.1 Register Descriptions

This section describes the I²C registers in detail.

NOTE

Reserved bits should always be written with the value they returned when read. That is, the register should be programmed by reading the value, modifying appropriate fields, and writing back the value. The return value of the reserved fields should not be assumed, even though the reserved fields return zero.

This note does not apply to the I²C data register (I2CDR).

11.3.1.1 I²C Address Register (I2CADR)

Figure 11-2 shows the I2CADR register, which contains the address to which the I²C interface responds when addressed as a slave. Note that this is not the address that is sent on the bus during the address-calling cycle when the I²C module is in master mode.

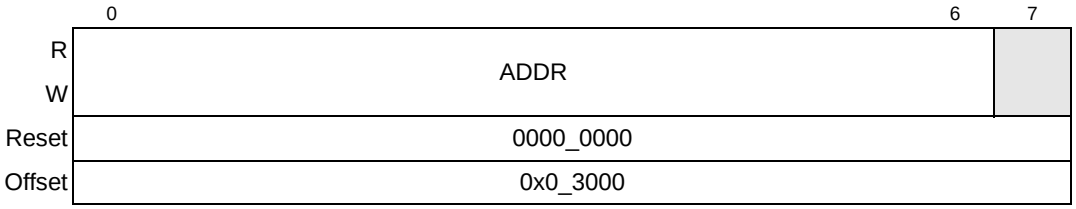


Figure 11-2. I²C Address Register (I2CADR)

Table 11-4 describes the bit settings of I2CADR.

Table 11-4. I2CADR Field Descriptions

Bits	Name	Description
0–6	ADDR	Slave address. Contains the specific slave address that is used by the I²C interface. Note that the default mode of the I²C interface is slave mode for an address match. Note that an address match is one of the conditions that can cause I2CSR[MIF] to be set, signaling an interrupt pending condition.
7	—	Reserved

11.3.1.2 I²C Frequency Divider Register (I2CFDR)

Figure 11-3 shows the bits of the I²C frequency divider register.

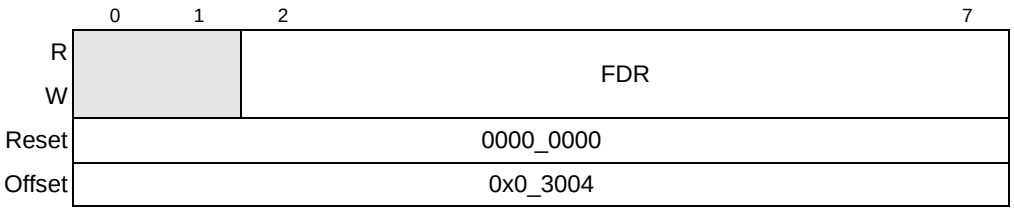


Figure 11-3. I²C Frequency Divider Register (I2CFDR)

Table 11-5 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) FDR Divider (Decimal) 0x00 288 0x16 12288 0x2B 1024 0x01 320 0x17 15360 0x2C 1280 0x02 384 0x18 18432 0x2D 1536 0x03 480 0x19 20480 0x2E 1792 0x04 576 0x1A 24576 0x2F 2048 0x05 640 0x1B 30720 0x30 2560 0x06 768 0x1C 36864 0x31 3072 0x07 960 0x1D 40960 0x32 3584 0x08 1152 0x1E 49152 0x33 4096 0x09 1280 0x1F 61440 0x34 5120 0x0A 1536 0x20 160 0x35 6144 0x0B 1920 0x21 192 0x36 7168 0x0C 2304 0x22 224 0x37 8192 0x0D 2560 0x23 256 0x38 10240 0x0E 3072 0x24 320 0x39 12288 0x0F 3840 0x25 384 0x3A 14336 0x10 4608 0x26 448 0x3B 16384 0x11 5120 0x27 512 0x3C 20480 0x12 6144 0x28 640 0x3D 24576 0x13 7680 0x29 768 0x3E 28672 0x14 9216 0x2A 896 0x3F 32768 0x15 10240

11.3.1.3 I²C Control Register (I2CCR)

Figure 11-4 shows the I²C control register.

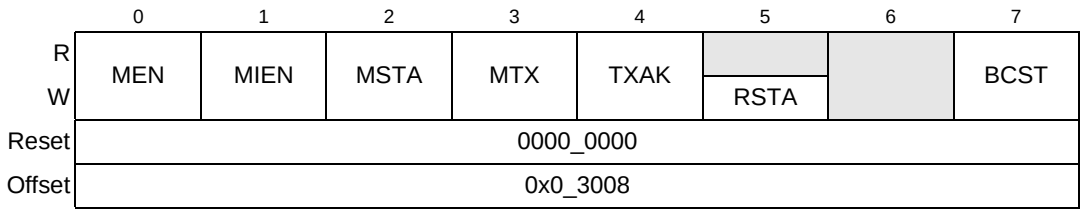


Figure 11-4. I²C Control Register (I2CCR)

Table 11-6 describes the bit settings of the I2CCR

Table 11-6. I2CCR Field Descriptions

Bits	Name	Description
0	MEN	Module enable. This bit controls the software reset of the I^2C module. 0 The module is reset and disabled. When low, the interface is held in reset but the registers can still be accessed. 1 The I^2C module is enabled. This bit must be set before any other control register bits have any effect. All I^2C registers for slave receive or master START can be initialized before setting this bit.
1	MIEN	Module interrupt enable 0 Interrupts from the I^2C module are disabled. This does not clear any pending interrupt conditions. 1 Interrupts from the I^2C module are enabled. An interrupt occurs provided I2CSR[MIF] is also set.
2	MSTA	Master/slave mode START 0 When this bit is changed from one to zero, a STOP condition is generated and the mode changes from master to slave. 1 Cleared without generating a STOP condition when the master loses arbitration. When this bit is changed from zero to one, a START condition is generated on the bus, and master mode is selected.
3	MTX	Transmit/receive mode select. This bit selects the direction of the master and slave transfers. When configured as a slave, this bit should be set by software according to I2CSR[SRW]. In master mode, the bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. The MTX bit is cleared when the master loses arbitration. 0 Receive mode 1 Transmit mode
4	TXAK	Transfer acknowledge. This bit specifies the value driven onto the SDA line during acknowledge cycles for both master and slave receivers. The value of this bit only applies when the I^2C 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, which means if a read is performed to I2CCR[RSTA], a zero value will be returned. 0 No START condition is generated 1 Generates repeat START condition
6	—	Reserved
7	BCST	Broadcast 0 Disables the broadcast accept capability 1 Enables the I^2C to accept broadcast messages at address zero

11.3.1.4 I^2C Status Register (I2CSR)

The status register, shown in Figure 11-5, is read only with the exception of the MIF and MAL bits, which can be cleared by software. The MCF and RXAK bits are set at reset; all other I2CSR bits are cleared on reset.

	0	1	2	3	4	5	6	7
R	MCF	MAAS	MBB	MAL	BCSTM	SRW	MIF	RXAK
W								
Reset	1000_0001							
Offset	0x0_300C							

Figure 11-5. I²C Status Register (I2CSR)

Table 11-7 describes the bit settings of the I2CSR.

Table 11-7. I2CSR Field Descriptions

Bits	Name	Description
0	MCF	Data transfer. When one byte of data is transferred, the bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. 0 Byte transfer in progress. MCF is cleared under the 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 this I²C drives an address of all 0s and broadcast mode is enabled.
5	SRW	Slave read/write. When MAAS is set, SRW indicates the value of the R/W command bit of the calling address, which is sent from the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave. This bit is valid only when both of the following conditions are true: • A complete transfer occurred and no other transfers have been initiated. • The I²C interface is configured as a slave and has an address match. By checking this bit, the processor can select slave transmit/receive mode according to the command of the master.

Table 11-7. I2CSR Field Descriptions (continued)

Bits	Name	Description
6	MIF	Module interrupt. The MIF bit is set when an interrupt is pending, causing a processor interrupt request (provided I2CCR[MIEN] is set). 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 I²C Data Register (I2CDR)

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

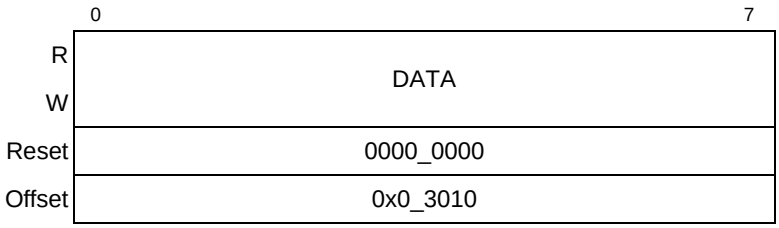


Figure 11-6. I²C Data Register (I2CDR)

Table 11-8 shows the bit descriptions for I2CDR.

Table 11-8. I2CDR Field Description

Bits	Name	Description
0–7	DATA	Transmission starts when an address and the R/W bit are written to the data register and the I²C interface performs as the master. A data transfer is initiated when data is written to the I2CDR. The most significant bit is sent first in both cases. In the master receive mode, reading the data register allows the read to occur, but also allows the I²C module to receive the next byte of data on the I²C interface. In slave mode, the same function is available after it is addressed. Note that the very first read is always a dummy read.

11.3.1.6 Digital Filter Sampling Rate Register (I2CDFSRR)

The digital filter sampling rate register (I2CDFSRR) is shown in Figure 11-7.

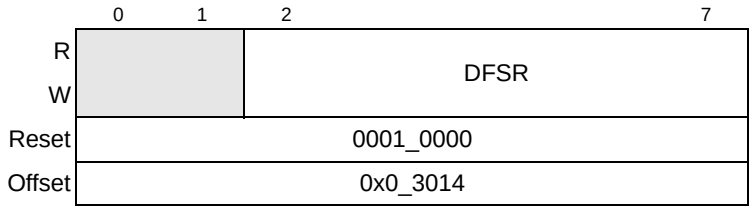


Figure 11-7. I²C Digital Filter Sampling Rate Register (I2CDFSRR)

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 I²C bus. The resulting sampling rate is calculated by dividing the platform (CCB clock) frequency by the non-zero value of DFSR. If I2CDFSRR is set to zero, the I²C bus sample points default to the reset divisor 0x10.

11.4 Functional Description

The I²C unit always performs as a slave receiver as a default, unless explicitly programmed to be a master or slave transmitter. After the boot sequencer has completed (when powered up in boot sequencer mode), the I²C interface will perform as a slave receiver.

11.4.1 Transaction Protocol

A standard I²C transfer consists of the following:

- START condition
- Slave target address transmission
- Data transfer
- STOP condition

Figure 11-8 shows the interaction of these four parts with the calling address, data byte, and new calling address components of the I²C protocol. The details of the protocol are described in the following subsections.

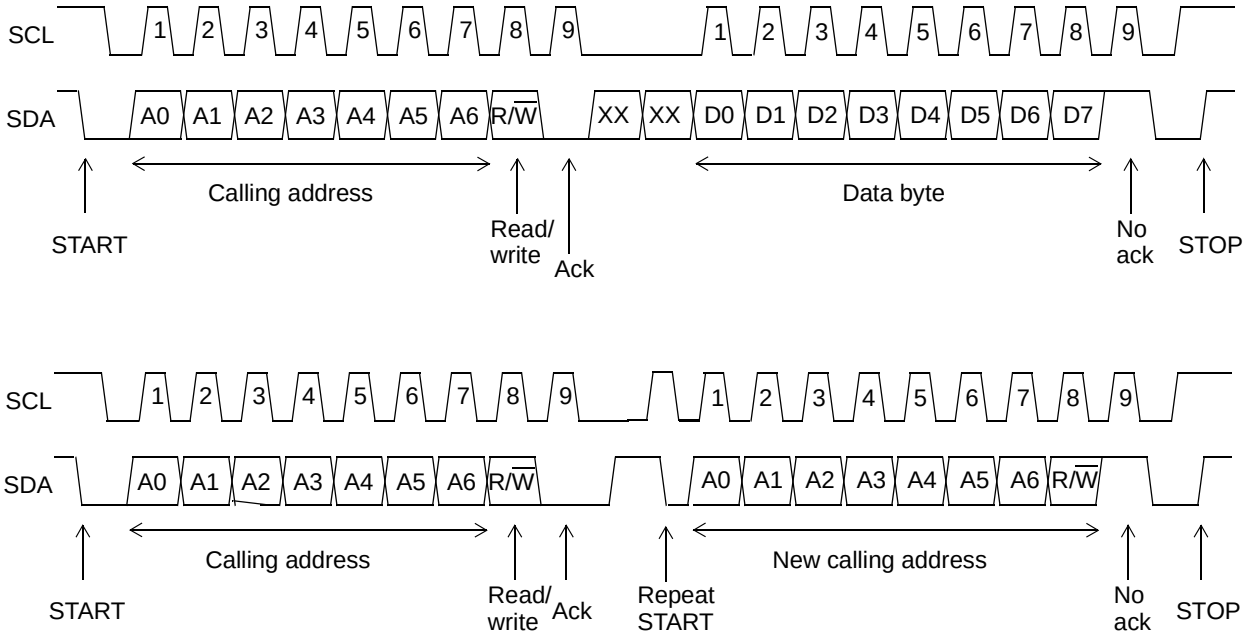


Figure 11-8. I^{2}C Interface Transaction Protocol

11.4.1.1 START Condition

When the I^{2}C 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 I2CCR[MSTA].

11.4.1.2 Slave Address Transmission

The first byte of data is transferred by the master immediately after the START condition is the slave address. This is a seven-bit calling address followed by a R/$\overline{W}$ bit, which indicates the direction of the data being transferred to the slave. Each slave in the system has a unique address. In addition, when the I^{2}C module is operating as a master, it must not transmit an address that is the same as its slave address. An I^{2}C 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 I²C module responds to a general call (broadcast) command when I2CCR[BCST] is set. A broadcast address is always zero; however the I²C module will not check the R/$\overline{W}$ bit. The second byte of the broadcast message is the master address. Because the second byte is automatically acknowledged by hardware, the receiver device software must verify that the broadcast message is intended for itself by reading the second byte of the message. If the master address is for another receiver device and the third byte is a write command, software can ignore the third byte during the broadcast. If the master address is for another receiver device and the third byte is a read command, software must write 0xFF to I2CDR with I2CCR[TXAK] = 1, so that it does not interfere with the data written from the addressed device.

Each data byte is 8 bits long. Data bits can be changed only while SCL is low and must be held stable while SCL is high, as shown in Figure 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 9th clock. Therefore, one complete data byte transfer takes 9 clock pulses. Several bytes can be transferred during a data transfer session.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA line for the master to generate a STOP or a START condition.

11.4.1.3 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.

11.4.1.4 STOP Condition

The master can terminate the transfer by generating a STOP condition to free the bus. A STOP condition is defined as a low-to-high transition of the SDA signal while SCL is high. For more information, see Figure 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 11.4.1.3, “Repeated START Condition,” the master can generate a START condition followed by a calling address without generating a STOP condition for the previous transfer. This is called a repeated START condition.

11.4.1.5 Protocol Implementation Details

The following sections give details of how aspects of the protocol are implemented in this I^2C module.

11.4.1.5.1 Transaction Monitoring—Implementation Details

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 an SDA rise occurs while SCL is high.
- Data transfers in progress are canceled when a STOP condition is detected or if there is a slave address mismatch. Cancellation of data transactions resets the clock module.
- The bus is detected to be busy upon the detection of a START condition, and idle upon the detection of a STOP condition.

11.4.1.5.2 Control Transfer—Implementation Details

The I^2C module 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 it is performing a START, STOP, or restart condition. Otherwise, the SDA output is held constant.

The SDA signal is pulled low when one or more of the following conditions are true in either master or slave mode:

- Master mode
  - Data bit (transmit)
  - Ack bit (receive)
  - START condition
  - STOP condition
  - Restart condition
- Slave mode
  - Acknowledging address match
  - Data bit (transmit)
  - Ack bit (receive)

The SCL signal corresponds to the internal SCL signal when one or more of the following conditions are true in either master or slave mode:

- Master mode
  - Bus owner
  - Lost arbitration
  - START condition
  - STOP condition
  - Restart condition begin
  - Restart condition end
- Slave mode
  - Address cycle
  - Transmit cycle
  - Ack cycle

11.4.1.6 Address Compare—Implementation Details

Address compare block determines if a slave has been properly addressed, either by its slave address or by the general broadcast address (which addresses all slaves). The three performed address comparisons are described as follows:

- Whether a broadcast message has been received, to update the I2CSR
- Whether the module has been addressed as a slave, to update the I2CSR and to generate an interrupt
- If the address transmitted by the current master matches the general broadcast address

11.4.2 Arbitration Procedure

The I²C interface is a true multiple-master bus that allows more than one master device to be connected on it. If two or more masters simultaneously try to control the bus, each master's clock synchronization procedure (including the I²C module) determines the bus clock—the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. A bus master loses arbitration if it transmits a logic 1 on SDA while another master transmits a logic 0. The losing masters immediately switch to slave-receive mode and stop driving the SDA line. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, the I²C unit sets the I2CSR[MAL] status bit to indicate the loss of arbitration and, as a slave, services the transaction if it is directed to itself.

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.

11.4.2.1 Arbitration Control

The arbitration control block controls the arbitration procedure of the master mode. A loss of arbitration occurs whenever the master detects a 0 on the external SDA line while attempting to drive a 1, tries to generate a START or restart at an inappropriate time, or detects an unexpected STOP request on the line.

In master mode, arbitration by the master is lost (and I2CSR[MAL] is set) under the following conditions:

- SDA samples low when the master drives high during an address or data-transmit cycle (transmit).
- SDA samples low when the master drives high during a data-receive cycle of the acknowledge (Ack) bit (receive).
- A START condition is attempted when the bus is busy.
- A repeated START condition is requested in slave mode.
- A start condition is attempted when the requesting device is not the bus owner
- Unexpected STOP condition detected

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

11.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices can hold SCL low after completion of a 1-byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

11.4.4 Clock Control

The clock control block handles requests from the clock signal for transferring and controlling data for multiple tasks.

A 9-cycle data transfer clock is requested for the following conditions:

- Master mode
  - Transmit slave address after START condition
  - Transmit slave address after restart condition
  - Transmit data
  - Receive data
- Slave mode
  - Transmit data
  - Receive data
  - Receive slave address after START or restart condition

11.4.4.1 Clock Synchronization

Due to the wire AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached. However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, the synchronized clock signal, SCL, is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time. When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

11.4.4.2 Input Synchronization and Digital Filter

The following sections describes the synchronizing of the input signals, and the filtering of the SCL and SDA lines in detail.

11.4.4.2.1 Input Signal Synchronization

The input synchronization block synchronizes the input SCL and SDA signals to the system clock and detects transitions of these signals.

11.4.4.2.2 Filtering of SCL and SDA Lines

The SCL and SDA inputs are filtered to eliminate noise. Three consecutive samples of the SCL and SDA lines are compared to a pre-determined sampling rate. If they are all high, the output of the filter is high. If they are all low, the output is low. If they are any combination of highs and lows, the output is whatever the value of the line was in the previous clock cycle.

The sampling rate is equal to a binary value stored in the frequency register I2CDFSRR. The duration of the sampling cycle is controlled by a down counter. This allows a software write to the frequency register to control the filtered sampling rate.

11.4.4.3 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven the SCL line low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period, then the resulting SCL bus signal low period is stretched.

11.4.5 Boot Sequencer Mode

If boot sequencer mode is selected on POR (by the settings on the LGPL3 and LGPL5 reset configuration signals, as described in Section 3.4.3.6, “Boot Sequencer Configuration”), the I^2C module communicates with one or more EEPROMs through the I^2C interface. The EEPROM(s) can be programmed to initialize one or more configuration registers of this integrated device.

The boot sequencer mode also supports an extension of the standard I^2C interface that uses more address bits to allow for EEPROM devices that have more than 256 bytes, and this extended addressing mode is selectable during POR with a different encoding on the LGPL3 and LGPL5 reset configuration signals (see Section 3.4.3.6, “Boot Sequencer Configuration”). In this mode, only one EEPROM device may be used, and the maximum number of registers is limited by the size of the EEPROM.

If the 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 CONT bit is cleared. If the last register is not detected before wrapping back to the first address, an error condition is detected. In other words, if the CONT bit for not cleared on the final 7 bytes, an error condition is detected, causing the device to hang and the $\overline{\text{HRESET\_REQ}}$ signal to assert externally. The I^2C module continues to read from the EEPROM as long as the continue (CONT) bit is set in the EEPROM. The CONT bit resides in the address/attributes field that is transferred from the EEPROM, as described in Section 11.4.5.1, “EEPROM Calling Address.” There should be no other I^2C traffic when the boot sequencer is active.

Note that as described in Section 3.4.3.6, “Boot Sequencer Configuration,” 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.

11.4.5.1 EEPROM Calling Address

The MPC8560 uses 0b101_0000 for the EEPROM calling address. The first EEPROM to be addressed must be programmed to respond to this address, or an error is generated. If more EEPROMs are used, they are addressed in sequential order.

11.4.5.2 EEPROM Data Format

The I²C module expects that a particular data format be used for data in the EEPROM. A preamble should be the first 3 bytes programmed into the EEPROM. It should have a value of 0xAA55AA. The I²C module checks to ensure that this preamble is correctly detected before proceeding further. Following the preamble, there should be a series of configuration registers (known as register preloads) programmed into the EEPROM. Each configuration register should be programmed according to a particular format, as shown in Figure 11-9. The first 3 bytes hold the attributes and address offset, as follows. The attributes contained are alternate configuration space (ACS), byte enables, and continue (CONT). The boot sequencer expects the address offset to be a 32-bit (word) offset, that is, the 2 low-order bits are not included in the boot sequencer command. For example, to access LAWBAR0 (byte offset of 0x00C08), the boot sequencer ADDR[0:17] should be set to 0x00302.

After the first 3 bytes, 4 bytes of data should hold the desired value of the configuration register, regardless of the size of the transaction. Byte enables should be asserted for any byte that will be written to the configuration register, and they should be asserted contiguously, creating a 1-, 2-, or 4-byte write to a register. The boot sequencer assumes that a big-endian address is stored in the EEPROM. In addition, byte enable bit 0 (bit 1 of the byte) corresponds to the most-significant byte of data (data[0:7]), and byte enable bit 3 (bit 4 of the byte) corresponds to the LSB of data (data[24:31]).

By setting ACS, an alternate configuration space address is prepended to the write request from the boot sequencer. Otherwise, CCSRBAR is prepended to the EEPROM address.

If CONT is cleared, the first 3 bytes, including ACS, the byte enables, and the address, must also be cleared. Also, the data contains the final cyclic redundancy check (CRC). A CRC-32 algorithm is used to check the integrity of the data. The polynomial used is:

$$1 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$$

CRC values are calculated using the above polynomial with a start value of 0xFFFF_FFFF and an XOR with 0x0000_0000. The CRC should cover all bytes stored in the EEPROM prior to the CRC. This includes the preamble, all register preloads, and the first 3 bytes of the last 7-byte preload (which should be all zeros). If a preamble or CRC fail is detected, the device hangs and the external $\overline{\text{HRESET\_REQ}}$ signal asserts. If there is a preamble fail, the boot sequencer may continue to pull I²C pins low until a hard reset occurs.

0	1	4	5	6	7
ACS	BYTE_EN		CONT	ADDR[0–1]	
ADDR[2–9]					
ADDR[10–17]					
DATA[0–7]					
DATA[8–15]					
DATA[16–23]					
DATA[24–31]					

Figure 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]								
. . .								

Figure 11-10. EEPROM Contents

ACS	BYTE_EN				1	ADDR[0–1]		Last Configuration Preload Command
ADDR[2–9]								
ADDR[10–17]								
DATA[0–7]								
DATA[8–15]								
DATA[16–23]								
DATA[24–31]								
0	0	0	0	0	0	0	0	End Command
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
CRC[0–7]								Cyclic Redundancy Check
CRC[8–15]								
CRC[16–23]								
CRC[24–31]								

Figure 11-10. EEPROM Contents (continued)

11.5 Initialization/Application Information

This section describes some programming guidelines recommended for the I²C interface. Figure 11-11 is a recommended flowchart for I²C interrupt service routines.

The I²C registers in this chapter are shown in big-endian format. If the system is in little-endian mode, software must swap the bytes appropriately. This appropriate byte swapping is needed as I²C registers are byte registers. Also, an **msync** assembly instruction must be executed after each I²C register read/write access to guarantee in-order execution.

The I²C controller does not guarantee its recovery from all illegal I²C bus activity. In addition, a malfunctioning device may hold the bus captive. A good programming practice is for software to rely on a watchdog timer to help recover from I²C bus hangs. The recovery routine should also handle the case when the status bits returned after an interrupt are not consistent with what was expected due to illegal I²C bus protocol behavior.

11.5.1 Initialization Sequence

A hard reset initializes all the I²C registers to their default states. The following initialization sequence initializes the I²C unit:

1. All I²C registers must be located in a cache-inhibited page.
2. Update I2CFDR[FDR] and select the required division ratio to obtain the SCL frequency from the CCB (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 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 MIF is set at any time, an I^2C interrupt is generated (provided interrupt reporting is enabled with I2CCR[MIEN] =1) so that the I^2C interrupt handler can 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 and an interrupt is generated to the processor if the interrupt function is enabled during the initialization sequence (I2CCR[MIEN] is set). In the interrupt handler, software must take the following steps:

1. Clear I2CSR[MIF]
2. Read the contents of the 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] must 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] must be polled rather than I2CSR[MCF] because MCF behaves differently when arbitration is lost. Note that interrupt or other bus conditions may be detected before the 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 I2C signals sufficient time to settle.

During slave-mode address cycles (I2CSR[MAAS] is set), I2CSR[SRW] should be read to determine the direction of the subsequent transfer and I2CCR[MTX] should be programmed accordingly. For slave-mode data cycles (MAAS is cleared), I2CSR[SRW] is not valid and I2CCR[MTX] must be read to determine the direction of the current transfer. See Section 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 (by setting the transmit acknowledge bit (I2CCR[TXAK])) before reading the next-to-last byte of data. At this time, the next-to-last byte of data has already been transferred on the I2C interface, so the last byte will not receive the data acknowledge (because I2CCR[TXAK] is set). For 1-byte transfers, a dummy read should be performed by the interrupt service routine (see Section 11.5.8, "Interrupt Service Routine Flowchart"). Before the interrupt service routine reads the last byte of data, a STOP condition must first be generated.

The I2C controller automatically generates a STOP if I2CCR[TXAK] is set. Therefore, I2CCR[TXAK] must be set before allowing the I2C module to receive the last data byte on the I2C bus. Eventually, I2CCR[TXAK] needs to be cleared again for subsequent I2C transactions. This can be accomplished when setting up the I2CCR for the next transfer.

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. This is accomplished by setting I2CCR[RSTA].

11.5.6 Generation of SCL When SDA Low

It is sometimes necessary to force the I2C module to become the I2C bus master out of reset and drive SCL (even though SDA may already be driven, which indicates that the bus is busy). This can occur when a system reset does not cause all I2C devices to be reset. Thus, SDA can be driven low by another I2C device while this I2C module is coming out of reset and will stay low indefinitely. The following procedure can be used to force this I2C module to generate SCL so that the device driving SDA can finish its transaction:

1. Disable the I2C module and set the master bit by setting I2CCR to 0x20
2. Enable the I2C module by setting I2CCR to 0xA0

3. Read the I2CDR
4. Return the I^2C module 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] is set, software should set the transmit/receive mode select bit (I2CCR[MTX]) according to the R/$\overline{W}$ command bit (I2CSR[SRW]). Writing to I2CCR clears MAAS automatically. MAAS is read as set only in the interrupt handler at the end of that address cycle where an address match occurred; interrupts resulting from subsequent data transfers clear MAAS. A data transfer can then be initiated by writing to I2CDR for slave transmits or dummy reading from I2CDR in slave-receive mode. The slave drives SCL low between byte transfers. SCL is released when the I2CDR is accessed in the required mode.

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] is set), the slave transmitter interrupt routine must clear I2CCR[MTX] to switch the slave from transmitter to receiver mode. A dummy read of I2CDR then releases SCL so that the master can generate a STOP condition. See Section 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 first test I2CSR[MAL] and software should clear it if it is set. See Section 11.4.2.1, “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), the interrupt service routine may need to set I2CCR[TXAK] when the next-to-last byte is to be accepted. It is recommended that an **msync** instruction follow each I^2C register read or write to guarantee in-order instruction execution.

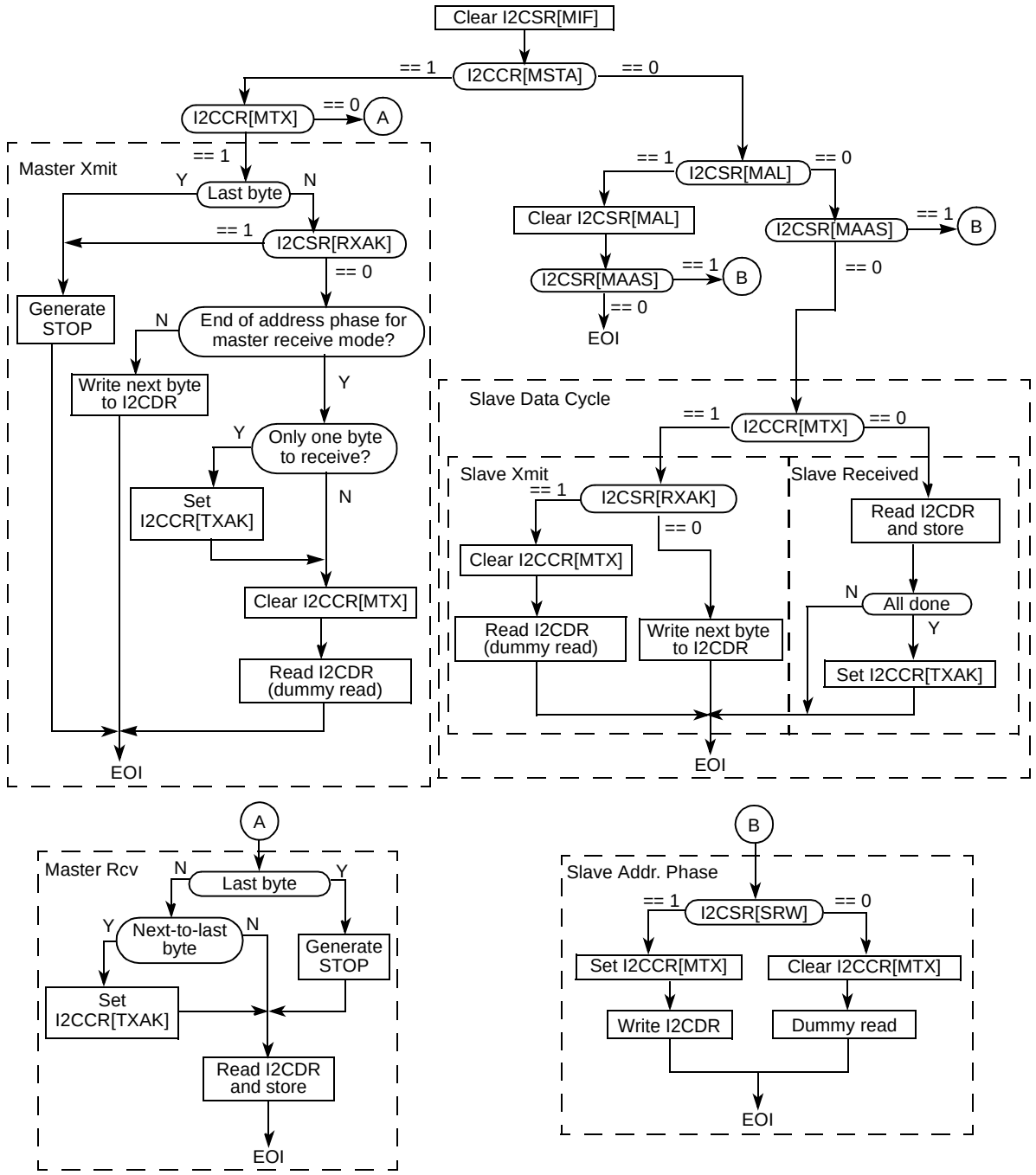


Figure 11-11. Example I²C Interrupt Service Routine Flowchart

Chapter 12
Local Bus Controller

This chapter describes the local bus controller (LBC) block. It describes the external signals and the memory-mapped registers as well as a functional description of the general-purpose chip-select machine (GPCM), SDRAM machine, and user-programmable machines (UPMs) of the LBC. Finally, it includes an initialization and applications information section with many specific examples of its use.

12.1 Introduction

Figure 12-1 is a functional block diagram of the LBC, which supports three interfaces: GPCM, UPM, and SDRAM controller.

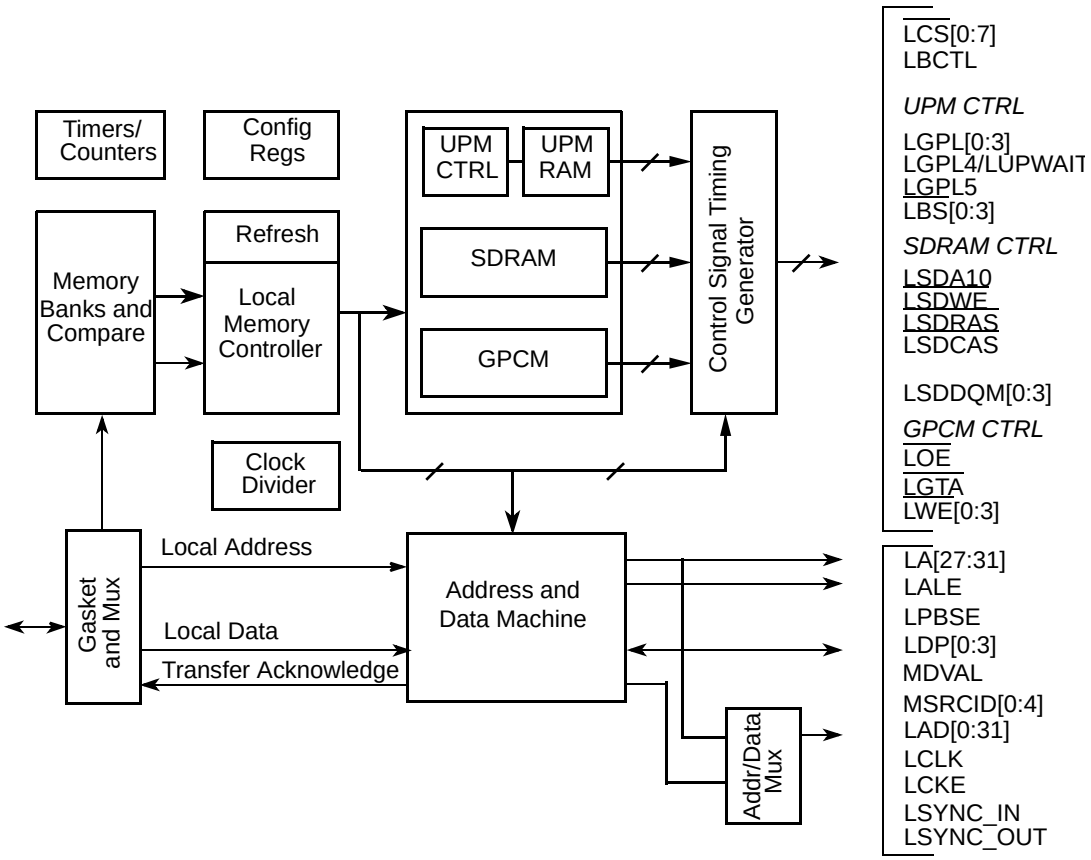


Figure 12-1. Local Bus Controller Block Diagram

12.1.1 Overview

The main component of the 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 GPCM, and up to three 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 LBC 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 LBC main features are as follows:

- Memory controller with eight memory banks
  - 34-bit address decoding with mask
  - Variable memory block sizes (32 Kbytes to 4 Gbytes)
  - Selection of control signal generation on a per-bank basis
  - Data buffer controls activated on a per-bank basis
  - 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
- SDRAM machine
  - Provides the control functions and signals for glueless connection to JEDEC-compliant SDRAM devices
  - Supports up to four concurrent open pages per device
  - Supports SDRAM port size of 32, 16, and 8 bits
  - 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 3-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-, and 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 LBC provides one GPCM, one SDRAM machine, and three UPMs for the local bus, with no restriction on how many of the eight banks (chip selects) can be programmed to operate with any given machine. When a memory transaction is dispatched to the LBC, 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, with the LBC in GPCM, SDRAM, or UPM mode, only one of the eight chip selects is active at any time for the duration of the transaction.

12.1.3.1 LBC Bus Clock and Clock Ratios

The LBC supports ratios of 2, 4, and 8 between the faster system (CCB) clock and the slower external bus clock (LCLK[0:2]). This ratio is software programmable through the clock ratio register (LCRR[CLKDIV]). In addition to establishing the frequency of the external local bus clock, CLKDIV also 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 pins, LCLK[0:2], 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 debug mode, the LBC provides the ID of a transaction source on external device pins. This mode is enabled on power-on reset, as described in Section 19.4.4, “Local Bus Interface Debug.” When placed in this mode, the 5-bit internal ID of the current transaction source appears on MSRCID[0:4] whenever valid address or data is available on the LBC external pins. The reserved value of 0x1F, which indicates invalid address or data, appears on the source ID pins at all other times. The combination of a valid source ID (any value except 0x1F) and the value of external address latch enable (LALE) and data valid (MDVAL) facilitate capturing useful debug data as follows:

- If a valid source ID is detected on MSRCID[0:4] and LALE is asserted, a valid full 32-bit address may be latched from LAD[0:31]. 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 LBC can enter a power-down mode when the system stops the internal (system) clock to the block by using a handshake protocol initiated by the DEVDISR[LBC] setting in the global utilities block. On entering power-down mode, the LBC places any SDRAM devices, if used, in self-refresh mode before the bus clock is stopped. The LBC also allows the DLL sufficient time to recover following the reapplication of the system clock.

12.1.5 References

- *MPC8260 PowerQUICC II Family Reference Manual*, Chapters 4, 6, and 10 (order no. MPC8260UM).

12.2 External Signal Descriptions

Table 12-1 contains a list of external signals related to the LBC and summarizes their function.

Table 12-1. Signal Properties—Summary

Name	Number of Signals	Direction	Function
LALE	1	Output	External address latch enable
$\overline{\text{LCS}}$	8	Output	Chip selects
$\overline{\text{LWE}}$/ LSDDQM/ $\overline{\text{LBS}}$	4	Output Output Output	GPCM mode: write enable SDRAM mode: byte lane data mask UPM mode: byte (lane) select
LSDA10/ LGPL0	1	Output Output	SDRAM mode: row address bit/command bit UPM mode: general-purpose line 0
$\overline{\text{LSDWE}}$/ LGPL1	1	Output Output	SDRAM mode: write enable UPM mode: general-purpose line 1
$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$/ LGPL2	1	Output Output Output	GPCM mode: output enable SDRAM mode: row address strobe UPM mode: general-purpose line 2
$\overline{\text{LSDCAS}}$/ LGPL3	1	Output Output	SDRAM mode: column address strobe UPM mode: general-purpose line 3
$\overline{\text{LGTA}}$/ LGPL4/ LUPWAIT/ 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
LGPL5	1	Output	UPM mode: general-purpose line 5
LBCTL	1	Output	Data buffer control
LA[27:31]	5	Output	Local bus non-multiplexed address lsb's
LAD[0:31]	32	Input/Output	Multiplexed address/data bus
LDP	4	Input/Output	Local bus data parity
LCKE	1	Output	Local bus clock enable
LCLK[0:2]	3	Output	Local bus clocks
LSYNC_IN	1	Input	DLL synchronize input
LSYNC_OUT	1	Output	DLL synchronize output
MDVAL	1	Output	In LBC debug mode: local bus data valid
MSRCID	5	Output	In LBC debug mode: local bus source ID

Table 12-2 shows the detailed external signal descriptions for the LBC.

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/Negated—LALE is asserted with the address at the beginning of each memory controller transaction. The number of cycles for which it is asserted is governed by the ORn[EAD] and LCRR[EADC] fields. The exact timing of the negation of LALE is controlled by the LBCR[AHD] field. Note that no other control signals are asserted during the assertion of LALE.
$\overline{\text{LCS}}$[0:7]	O	Chip selects. Eight chip selects are provided 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.
$\overline{\text{LWE}}$[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 BR*n*[PS]), all four signals are defined. For a 16-bit port size, only bits 0–1 are defined; and for an 8-bit port size, bit 0 is the only defined signal. The least significant address bits of each access also determine which byte lanes are considered valid for a given data transfer.	
		State Meaning	Asserted/Negated—For GPCM operation, $\overline{\text{LWE}}$[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, “RAM Array,” for programming details about $\overline{\text{LBS}}$[0:3].
		Timing	Assertion/Negation—See Section 12.4.2, “General-Purpose Chip-Select Machine (GPCM),” for details regarding the timing of $\overline{\text{LWE}}$[0:3].
LSDA10/ LGPL0	O	SDRAM A10/General-purpose line 0	
		State Meaning	Asserted/Negated—For SDRAM accesses, represents address bit 10. When the row address is driven, it drives the value of address bit 10. When the column address is driven, it forms part of the SDRAM command. One of six general-purpose signals when in UPM mode; it drives a value programmed in the UPM array.
$\overline{\text{LSDWE}}$/ LGPL1	O	SDRAM write enable/General-purpose line 1	
		State Meaning	Asserted/Negated—Should be connected to the SDRAM device WE input. Acts as the SDRAM write enable when accessing SDRAM. One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.
$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$/ LGPL2	O	GPCM output enable/SDRAM $\overline{\text{RAS}}$/General-purpose line 2	
		State Meaning	Asserted/Negated—Controls the output buffer of memory when accessing memory/devices in GPCM mode. For SDRAM accesses, it is the row address strobe ($\overline{\text{RAS}}$). One of six general-purpose lines when in UPM mode; it drives a value programmed in the UPM array.

Table 12-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
LSDCAS/ LGPL3	O	SDRAM CAS/General-purpose line 3	
		State Meaning	Asserted/Negated—In SDRAM mode, drives the column address strobe (CAS). One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.
LGTA/ LGPL4/ LUPWAIT/ LPBSE	I/O	GPCM transfer acknowledge/General-purpose line 4/UPM wait/parity byte select	
		State Meaning	Asserted/Negated—Input in GPCM mode used for transaction termination. It may also be configured as one of six general-purpose output signals when in UPM mode or as an input to force the UPM controller to wait for the memory/device. When configured as LPBSE, it disables any use in GPCM or UPM modes. Because systems that use read-modify-write parity require an additional memory device, they must generate a byte-select like a normal data device. ANDing LBS[0:3] through external logic to achieve the logical function of this byte-select adds a delay to the byte-select path that can affect memory access timing. The LBC 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.
LGPL5	O	General-purpose line 5	
		State Meaning	Asserted/Negated—One of six general-purpose signals when in UPM mode, and drives a value programmed in the UPM array.
LBCTL	O	Data buffer control. The memory controller activates LBCTL for the local bus when a GPCM- or UPM-controlled bank is accessed. Access to an SDRAM machine-controlled bank does not activate the buffer control. Buffer control is disabled by setting OR*n*[BCTLD].	
		State Meaning	Asserted/Negated—The LBCTL pin normally functions as a write/read control for a bus transceiver connected to the LAD lines. Note that an external data buffer must not drive the LAD lines in conflict with the LBC when LBCTL is high, because LBCTL remains high after reset and during address phases.
LA[27:31]	O	Local bus nonmultiplexed address lsbs. All bits driven on LA[27:31] are defined for 8-bit port sizes. For 32-bit port sizes, LA[30:31] are don't cares; for 16-bit port sizes LA31 is a don't care.	
		State Meaning	Asserted/Negated—Although the LBC shares an address and data bus, up to five lsbs 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 lsbs 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.
LAD[0:31]	I/O	Multiplexed address/data bus. For configuration of a port size in BR*n*[PS] as 32 bits, all of LAD[0:31] must be connected to the external RAM data bus, with LAD[0:7] occupying the most significant byte lane (at address offset 0). For a port size of 16 bits, LAD[0:7] connect to the most significant byte lane (at address offset 0), while LAD[8:15] connect to the least-significant byte lane (at address offset 1); LAD[16:31] are unused for 16-bit port sizes. For a port size of 8 bits, only LAD[0:7] are connected to the external RAM.	
		State Meaning	Asserted/Negated—LAD[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. Drives and receives the data parity corresponding with the data phases on LAD[0:31].	
		State Meaning	Asserted/Negated—During write accesses, a parity bit is generated for each 8 bits of LAD[0:31], such that LDP0 is even/odd parity for LAD[0:7], while LDP3 is even/odd parity for LAD[24:31]. Unused byte lanes for port sizes less than 32 bits have undefined parity.
		Timing	Assertion/Negation—Drive and receive the data parity corresponding with the data phases on LAD[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]. LDP[0:3] change impedance in concert with LAD[0:31].
LCKE	O	Local bus clock enable	
		State Meaning	Asserted/Negated—LCKE is the bus clock enable signal (CKE) for JEDEC-standard SDRAM devices. Asserted during normal SDRAM operation.
LCLK[0:2]	O	Local bus clocks	
		State Meaning	Asserted/Negated—LCLK[0:2] drive an identical bus clock signal for distributed loads. If the LBC DLL is enabled (see LCRR[DBYP], Figure 12-19 on page 12-31), the bus clock phase is shifted earlier than transitions on other LBC signals (such as LAD[0:31] and LCS*n*) 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[0:2] 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/Negated—For a read, MDVAL asserts for one 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 during the final cycle for which the current write data on LAD[0:31] is valid. During burst transfers, MDVAL asserts for each data beat.
		Timing	Assertion/Negation—Valid only while the LBC is in system debug mode. In debug mode, MDVAL asserts when the LBC generates a data transfer acknowledge.
MSRCID[0:4]	O	Local bus source ID (LBC debug mode only). In debug mode, all MSRCID[0:4] pins are driven high unless MSRCID[0:4] is driving a debug source ID for identifying the internal system device controlling the LBC.	
		State Meaning	Asserted/Negated—Remain high until the last bus cycle of the assertion of LALE, in which case the source ID of the address is indicated, or until MDVAL is asserted, in which case the source ID relating to the data transfer is indicated. In case of address debug, MSRCID[0:4] is valid only when the address on LAD[0:31] consists of all physical address bits—with optional padding—for reconstructing the system address presented to the LBC. For example, MSRCID[0:4] is valid only during $\overline{\text{CAS}}$ phases of SDRAM accesses, because the column, bank select, and (normally unused) row address bits are all present on LAD[0:31] during a $\overline{\text{CAS}}$ cycle

12.3 Memory Map/Register Definition

Table 12-3 shows the memory mapped registers of the LBC. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 12-3. Local Bus Controller Memory Map

Address Offset	Use	Access	Reset	Section/Page
0x0_5000	BR0—Base register 0	R/W	0x0000_*nn*01[1]	12.3.1.1/12-10
0x0_5008	BR1—Base register 1		0x0000_0000	
0x0_5010	BR2—Base register 2			
0x0_5018	BR3—Base register 3			
0x0_5020	BR4—Base register 4			
0x0_5028	BR5—Base register 5			
0x0_5030	BR6—Base register 6			
0x0_5038	BR7—Base register 7			
0x0_5004	OR0—Options register 0	R/W	0x0000_0FF7	12.3.1.2/12-11
0x0_500C	OR1—Options register 1		0x0000_0000	
0x0_5014	OR2—Options register 2			
0x0_501C	OR3—Options register 3			
0x0_5024	OR4—Options register 4			
0x0_502C	OR5—Options register 5			
0x0_5034	OR6—Options register 6			
0x0_503C	OR7—Options register 7			
0x0_5068	MAR—UPM address register	R/W	0x0000_0000	12.3.1.3/12-17
0x0_5070	MAMR—UPMA mode register	R/W	0x0000_0000	12.3.1.4/12-18
0x0_5074	MBMR—UPMB mode register	R/W	0x0000_0000	12.3.1.4/12-18
0x0_5078	MCMR—UPMC mode register	R/W	0x0000_0000	12.3.1.4/12-18
0x0_5084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	12.3.1.5/12-20
0x0_5088	MDR—UPM data register	R/W	0x0000_0000	12.3.1.6/12-21
0x0_5094	LSDMR—SDRAM mode register	R/W	0x0000_0000	12.3.1.7/12-21
0x0_50A0	LURT—UPM refresh timer	R/W	0x0000_0000	12.3.1.8/12-23
0x0_50A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	12.3.1.9/12-24
0x0_50B0	LTESR—Transfer error status register	Read/Bit reset	0x0000_0000	12.3.1.10/12-25
0x0_50B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	12.3.1.11/12-26
0x0_50B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	12.3.1.12/12-27
0x0_50BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	12.3.1.13/12-28
0x0_50C0	LTEAR—Transfer error address register	R/W	0x0000_0000	12.3.1.14/12-29
0x0_50D0	LBCR—Configuration register	R/W	0x0000_0000	12.3.1.15/12-30
0x0_50D4	LCRR—Clock ratio register	R/W	0x8000_0008	12.3.1.16/12-31

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

12.3.1 Register Descriptions

This section provides a detailed description of the LBC 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 should be written to reserved bits of defined registers, as writing ones can have unpredictable results in some cases.

Bits designated as write-one-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

12.3.1.1 Base Registers (BR0–BR7)

The base registers (BR*n*), shown in Figure 12-2, contain the base address and address types for each memory bank. The memory controller uses this information to compare the address bus value with the current address accessed. Each register (bank) includes a memory attribute and selects the machine for memory operation handling. Note that after system reset, BR0[V] is set, BR1[V]–BR7[V] are cleared, and the value of BR0[PS] reflects the initial port size configured by the boot ROM location pins.

	0–16	17–18	19–20	21–22	23	24–26	27	28–29	30	31
R W	BA	—	PS	DECC	WP	MSEL		ATOM		V
Reset	0000_0000_0000_0000_000P_S000_0000_0000/1 [1]									
Offset	0x0_5000 (BR0); 0x0_5008 (BR1); 0x0_5010 (BR2); 0x0_5018 (BR3); 0x0_5020 (BR4); 0x0_5028 (BR5); 0x0_5030 (BR6); 0x0_5038 (BR7);									

[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*n*)

Table 12-4 describes BR*n* fields.

Table 12-4. BR*n* Field Descriptions

Bits	Name	Description
0–16	BA	Base address. The upper 17 bits of each base register are compared to the address on the address bus to determine if the bus master is accessing a memory bank controlled by the memory controller. Used with the address mask bits OR*n*[AM].
17–18	—	Reserved
19–20	PS	Port size. Specifies the port size of this memory region. For BR0, PS is configured from the 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

Table 12-4. BR*n* Field Descriptions (continued)

Bits	Name	Description
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 $\overline{\text{LCS}n}$ on write cycles to this memory bank. 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.
24–26	MSEL	Machine select. Specifies the machine to use for handling memory operations. 000 GPCM (reset value) 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 BR*n* and OR*n* pair are valid. $\overline{\text{LCS}n}$ does not assert unless V is set (an access to a region that has no valid bit set may cause a bus time-out). After a system reset, only BR0[V] is set. 0 This bank is invalid. 1 This bank is valid.

12.3.1.2 Option Registers (OR0–OR7)

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

- GPCM mode
- UPM mode
- SDRAM mode

The OR*n* 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*n*[XAM,AM]) mask up to 19 corresponding BR*n*[BA,XBA] fields. 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 to 4 Gbytes.

Table 12-5. Memory Bank Sizes in Relation to Address Mask

Bits 17–18	AM	Memory Bank Size
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 Mbytes
11	1111_1100_0000_0000_0	64 Mbytes
11	1111_1110_0000_0000_0	32 Mbytes
11	1111_1111_0000_0000_0	16 Mbytes
11	1111_1111_1000_0000_0	8 Mbytes
11	1111_1111_1100_0000_0	4 Mbytes
11	1111_1111_1110_0000_0	2 Mbytes
11	1111_1111_1111_0000_0	1 Mbyte
11	1111_1111_1111_1000_0	512 Kbytes
11	1111_1111_1111_1100_0	256 Kbytes
11	1111_1111_1111_1110_0	128 Kbytes
11	1111_1111_1111_1111_0	64 Kbytes
11	1111_1111_1111_1111_1	32 Kbytes

12.3.1.2.2 Option Registers (OR*n*)—GPCM Mode

Figure 12-3 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects the GPCM machine.

	0–16	17–18	19	20	21–22	23	24–27	28	29	30	31
R W	AM	—	BCTLD	CSNT	ACS	XACS	SCY	SETA	TRLX	EHTR	EAD
Reset	0000_0000_0000_0000_0000_1111_1111_0111 [1] (OR0); 0000_0000_0000_0000_0000_0000_0000_0000 (all others)										
Offset	0x0_5004 (OR0); 0x0_500C (OR1); 0x0_5014 (OR2); 0x0_501C (OR3); 0x0_5024 (OR4); 0x0_502C (OR5); 0x0_5034 (OR6); 0x0_503C (OR7);										

[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 (OR*n*) in GPCM Mode

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

Table 12-6. OR*n*—GPCM Field Descriptions

Bits	Name	Description
0–16	AM	GPCM address mask. Masks corresponding BR*n* bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked. 1 Corresponding address bits are used in the comparison between base and transaction addresses.
17–18	—	Reserved
19	BCTLD	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.
20	CSNT	Chip select negation time. Determines when $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated during an external memory write access handled by the GPCM, provided that ACS ≠ 00 (when ACS = 00, only $\overline{\text{LWE}}$ is affected by the setting of CSNT). This helps meet address/data hold times for slow memories and peripherals. 0 $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated normally. 1 $\overline{\text{LCS}n}$ and $\overline{\text{LWE}}$ are negated earlier depending on the value of LCRR[CLKDIV].

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

Table 12-6. OR*n*—GPCM Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>21–22</td><td>ACS</td><td>Address to chip-select setup. Determines the delay of the $\overline{\text{LCS}n}$ assertion relative to the address change when the external memory access is handled by the GPCM. At system reset, OR0[ACS] = 11.
<table>
<tr><th>LCRR[CLKDIV]</th><th>Value</th><th>Meaning</th></tr>
<tr><td rowspan="2">x</td><td>00</td><td>$\overline{\text{LCS}n}$ is output at the same time as the address lines. Note that this overrides the value of CSNT such that CSNT=0.</td></tr>
<tr><td>01</td><td>Reserved.</td></tr>
<tr><td rowspan="2">2</td><td>10</td><td>$\overline{\text{LCS}n}$ is output a half bus clock cycle after the address lines.</td></tr>
<tr><td>11</td><td>$\overline{\text{LCS}n}$ is output a half bus clock cycle after the address lines.</td></tr>
<tr><td rowspan="2">4 or 8</td><td>10</td><td>$\overline{\text{LCS}n}$ is output a quarter bus clock cycle after the address lines.</td></tr>
<tr><td>11</td><td>$\overline{\text{LCS}n}$ 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 $\overline{\text{LCS}n}$ assertion relative to the address change when the external memory access is handled by the GPCM. After a system reset, OR0[XACS] = 1.
0 Address to chip-select setup is determined by ORx[ACS] and LCRR[CLKDIV].
1 Address to chip-select setup is extended (see Table 12-23 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 $\overline{\text{LGTA}}$ earlier to terminate the access.
1 Access is terminated externally by asserting the $\overline{\text{LGTA}}$ external pin. (Only $\overline{\text{LGTA}}$ can terminate the access).</td></tr>
<tr><td>29</td><td>TRLX</td><td>Timing relaxed. Modifies the settings of timing parameters for slow memories or peripherals.
0 Normal timing is generated by the GPCM.
1 Relaxed timing on the following parameters:
• Adds an additional cycle between the address and control signals (only if ACS ≠ 00)
• Doubles the number of wait states specified by SCY, providing up to 30 wait states
• Works in conjunction with EHTR to extend hold time on read accesses
• $\overline{\text{LCS}n}$ (only if ACS ≠ 00) and $\overline{\text{LWE}}$ signals are negated one cycle earlier during writes.</td></tr>
</table>

Table 12-6. OR*n*—GPCM Field Descriptions (continued)

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

12.3.1.2.3 Option Registers (OR*n*)—UPM Mode

Figure 12-4 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects a UPM machine.

	0–18	19	20–22	23	24–28	29	30	31
R W	AM	BCTLD		BI		TRLX	EHTR	EAD
Reset	0000_0000_0000_0000_0000_0000_0000_0000							
Offset	0x0_5004 (OR0); 0x0_500c (OR1); 0x0_5014 (OR2); 0x0_501c (OR3); 0x0_5024 (OR4); 0x0_502c (OR5); 0x0_5034 (OR6); 0x0_503c (OR7);							

Figure 12-4. Option Registers (OR*n*) in UPM Mode

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

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

Bits	Name	Description
0–16	AM	UPM address mask. Masks corresponding BR*n* bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 0 Corresponding address bits are masked. 1 The corresponding address bits are used in the comparison with address pins.
17–18	—	Reserved

Table 12-7. OR*n*—UPM Field Descriptions (continued)

Bits	Name	Description
19	BCTLD	Buffer control disable. Disables assertion of LBCTL during access to the current memory bank. 0 LBCTL is asserted upon access to the current memory bank. 1 LBCTL is not asserted upon access to the current memory bank.
20–22	—	Reserved
23	BI	Burst inhibit. Indicates if this memory bank supports burst accesses. 0 The bank supports burst accesses. 1 The bank does not support burst accesses. The selected UPM executes burst accesses as a series of single accesses.
24–28	—	Reserved
29	TRLX	Timing relaxed. Works in conjunction with EHTR to extend hold time on read accesses.
30	EHTR	Extended hold time on read accesses. Indicates with TRLX how many cycles are inserted between a read access from the current bank and the next access.
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).

TRLX	EHTR	Meaning
0	0	The memory controller generates normal timing. No additional cycles are inserted.
0	1	1 idle clock cycle is inserted.
1	0	4 idle clock cycles are inserted.
1	1	8 idle clock cycles are inserted.

12.3.1.2.4 Option Registers (OR*n*)—SDRAM Mode

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

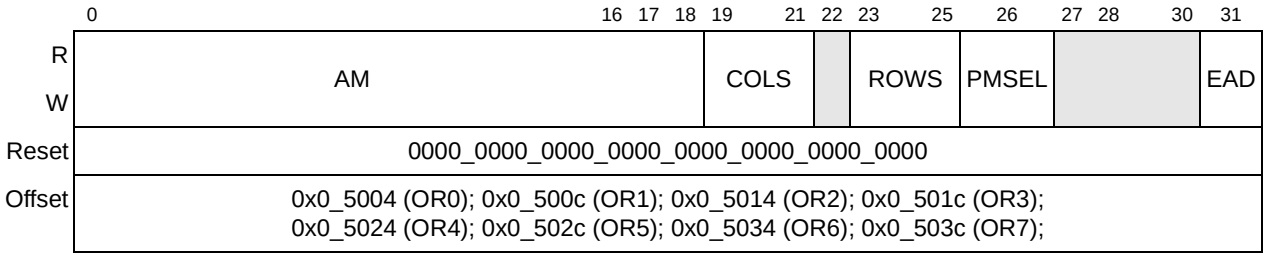


Figure 12-5. Option Registers (OR*n*) in SDRAM Mode

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

Table 12-8. OR*n*—SDRAM Field Descriptions

Bits	Name	Description
0–16	AM	SDRAM address mask. Masks corresponding BR*n* bits. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map. 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	—	Reserved
19–21	COLS	Number of column address lines. Sets the number of column address lines in the SDRAM device. 000 7 100 11 001 8 101 12 010 9 110 13 011 10 111 14
22	—	Reserved
23–25	ROWS	Number of row address lines. Sets the number of row address lines in the SDRAM device. 000 9 100 13 001 10 101 14 010 11 110 15 011 12 111 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.
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 shows the fields of the UPM memory address register (MAR).

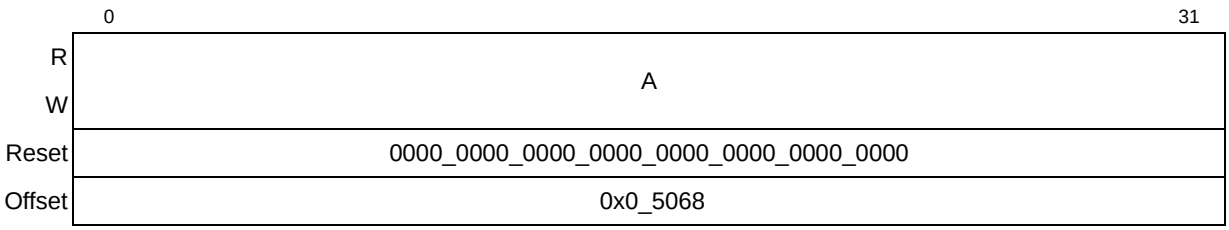


Figure 12-6. UPM Memory Address Register (MAR)

Table 12-9 describes the MAR field.

Table 12-9. MAR Field Description

Bits	Name	Description
0–31	A	Address that can be output to the address signals under control of the AMX bits in the UPM RAM word.

12.3.1.4 UPM Mode Registers (M*x*MR)

The UPM machine mode registers (MAMR, MBMR and MCMR), shown in Figure 12-7, contain the configuration for the three UPMs.

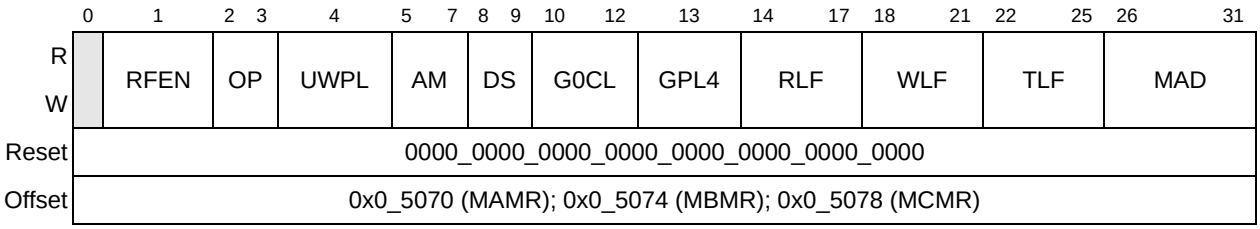


Figure 12-7. UPM Mode Registers (M*x*MR)

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 can be provided, even if UPMB and/or UPMC have their RFEN bit set. 0 Refresh services are not required 1 Refresh services are required
2–3	OP	Command opcode. Determines the command executed by the UPM*n* when a memory access hits a UPM assigned bank. See Section 12.4.4.2, “Programming the UPMs,” for important programming considerations. 00 Normal operation 01 Write to UPM array. On the next memory access that hits a UPM assigned bank, write the contents of the MDR into the RAM location pointed to by MAD. After the access, MAD is automatically incremented. 10 Read from UPM array. On the next memory access that hits a UPM assigned bank, read the contents of the RAM location pointed to by MAD into the MDR. After the access, MAD is automatically incremented. 11 Run pattern. On the next memory access that hits a UPM assigned bank, run the pattern written in the RAM array. The pattern run starts at the location pointed to by MAD and continues until the LAST bit is set in the RAM word.
4	UWPL	LUPWAIT polarity active low. Sets the polarity of the LUPWAIT pin when in UPM mode. 0 LUPWAIT is active high. 1 LUPWAIT is active low.

Table 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 UPMn. The disable timer is turned on by the TODT bit in the RAM array word, and when expired, the UPMn allows the machine access to handle a memory pattern to the same bank. Accesses to a different bank by the same UPMn is also allowed. To avoid conflicts between successive accesses to different banks, the minimum pattern in the RAM array for a request serviced, should not be shorter than the period established by DS.
00 1-bus clock cycle disable period
01 2-bus clock cycle disable period
10 3-bus clock cycle disable period
11 4-bus clock cycle disable period</td></tr>
<tr><td>10–12</td><td>G0CL</td><td>General line 0 control. Determines which logical address line can be output to the LGPL0 pin when the UPMn is selected to control the memory access.
000 A12
001 A11
010 A10
011 A9
100 A8
101 A7
110 A6
111 A5</td></tr>
<tr><td>13</td><td>GPL4</td><td>LGPL4 output line disable. Determines how the LGPL4/LUPWAIT pin is controlled by the corresponding bits in the UPMn array. See Table 12-30 on page 12-68.
<table>
<tr><th rowspan="2">Value</th><th rowspan="2">LGPL4/LUPWAIT Pin Function</th><th colspan="2">Interpretation of UPM Word Bits</th></tr>
<tr><th>G4T1/DLT3</th><th>G4T3/WAEN</th></tr>
<tr><td>0</td><td>LGPL4 (output)</td><td>G4T1</td><td>G4T3</td></tr>
<tr><td>1</td><td>LUPWAIT (input)</td><td>DLT3</td><td>WAEN</td></tr>
</table></td></tr>
</table>

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

Bits	Name	Description
14–17	RLF	Read loop field. Determines the number of times a loop defined in the UPM*n* will be executed for a burst- or single-beat read pattern or when M*x*MR[OP] = 11 (RUN command) 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15
18–21	WLF	Write loop field. Determines the number of times a loop defined in the UPM*n* will be executed for a burst- or single-beat write pattern. 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15
22–25	TLF	Refresh loop field. Determines the number of times a loop defined in the UPM*n* will be executed for a refresh service pattern. 0000 16 0001 1 0010 2 0011 3 ... 1110 14 1111 15
26–31	MAD	Machine address. RAM address pointer for the command executed. This field is incremented by 1 each time the UPM is accessed, and the OP field is set to WRITE or READ. Address range is 64 words per UPM*n*.

12.3.1.5 Memory Refresh Timer Prescaler Register (MRTPR)

The refresh timer prescaler register (MRTPR), shown in Figure 12-8, is used to divide the system clock to provide the SDRAM and UPM refresh timers clock.

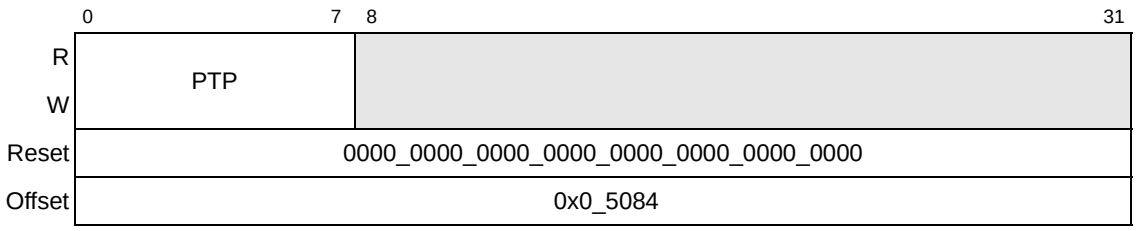


Figure 12-8. Memory Refresh Timer Prescaler Register (MRTPR)

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 0000_0000, which represents the maximum divider of 256.
8–31	—	Reserved

12.3.1.6 UPM Data Register (MDR)

The memory data register (MDR), shown in Figure 12-9, 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.

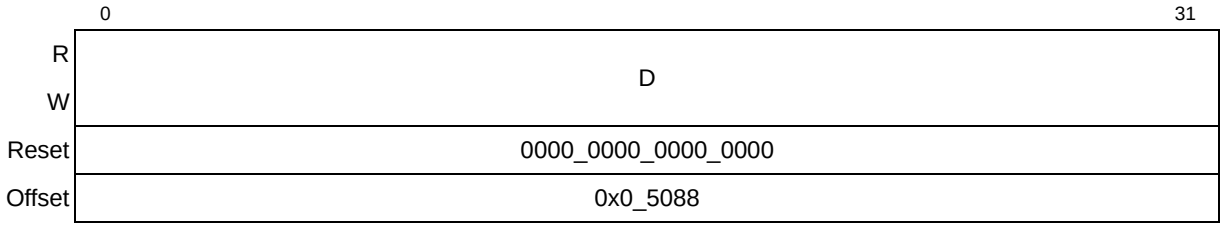


Figure 12-9. UPM Data Register (MDR)

Table 12-12 describes MDR[D].

Table 12-12. MDR Field Description

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 (M*x*MR[OP] = 01 or M*x*MR[OP] = 10).

12.3.1.7 SDRAM Machine Mode Register (LSDMR)

The local bus SDRAM mode register (LSDMR), shown in Figure 12-10, is used to configure operations pertaining to SDRAM.

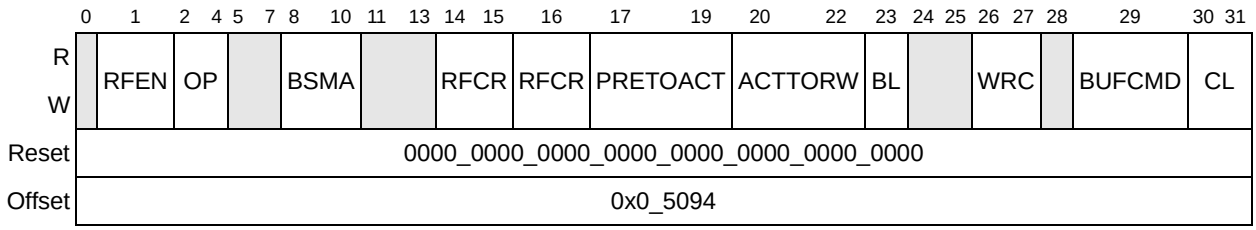


Figure 12-10. SDRAM Machine Mode Register (LSDMR)

Table 12-12 describes LSDMR fields.

Table 12-13. LSDMR Field Descriptions

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>—</td><td>Reserved</td></tr>
<tr><td>1</td><td>RFEN</td><td>Refresh enable. Indicates that the SDRAM requires refresh services.
0 Refresh services are not required.
1 Refresh services are required.</td></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. Note that only 4-bank SDRAMs are supported.
000 LA12:LA13 100 LA16:LA17
001 LA13:LA14 101 LA17:LA18
010 LA14:LA15 110 LA18:LA19
011 LA15:LA16 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 100 6 clocks
001 3 clocks 101 7 clocks
010 4 clocks 110 8 clocks
011 5 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 8 100 4
001 1 101 5
010 2 110 6
011 3 111 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 8 100 4 001 Reserved 101 5 010 2 110 6 011 3 111 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 $\overline{\text{LCS}n}$, 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 $\overline{\text{LCS}n}$ are asserted for the number of cycles specified by LCRR[BUFCMDC].
30–31	CL	CAS latency. Defines the timing for first read data after SDRAM samples a column address. 00 Extended CAS latency. According to LCRR[ECL]. See Table 12-22 on page 12-31. 01 1 10 2 11 3

12.3.1.8 UPM Refresh Timer (LURT)

The UPM refresh timer (LURT), shown in Figure 12-11, 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.

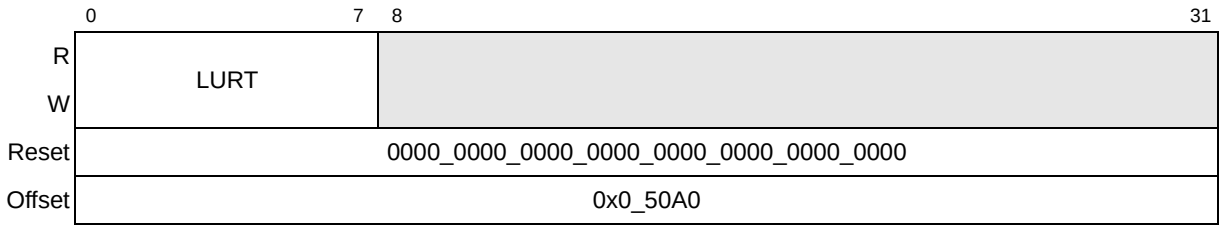


Figure 12-11. UPM Refresh Timer (LURT)

Table 12-14 describes LURT fields.

Table 12-14. LURT Field Descriptions

Bits	Name	Description
0–7	LURT	UPM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation: $TimerPeriod = \frac{LURT}{\left(\frac{Fsystemclock}{MRTPR[PTP]}\right)}$ Example: For a 266-MHz system clock and a required service rate of 15.6 µs, given MRTPR[PTP] = 32, the LURT value should be 128 decimal. 128/(266 MHz/32) = 15.4 µs, which is less than the required service period of 15.6 µs. Note that the reset value (0x00) sets the maximum period to 256 x MRTPR[PTP] system clock cycles.
8–31	—	Reserved

12.3.1.9 SDRAM Refresh Timer (LSRT)

The SDRAM refresh timer (LSRT), shown in Figure 12-12, 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 qualifying banks generate a bank staggering auto-refresh request using the SDRAM machine.

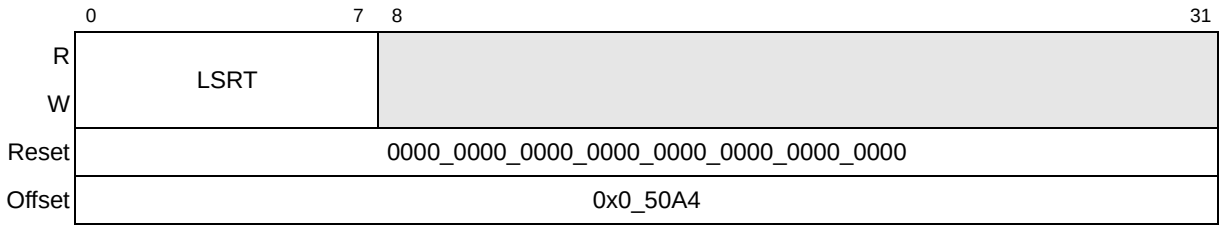


Figure 12-12. LSRT SDRAM Refresh Timer (LSRT)

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: $TimerPeriod = \frac{LSRT}{\left(\frac{Fsystemclock}{MRTPR[PTP]}\right)}$ **Example:** For a 266-MHz system clock and a required service rate of 15.6 µs, given PTP = 32, the LSRT value should be 128 decimal. 128/(266 MHz/32) = 15.4 µs, which is less than the required service period of 15.6 µs. Note that the reset value, 0x00, sets the maximum period to 256 × MRTPR[PTP] system clock cycles.
8–31	—	Reserved

12.3.1.10 Transfer Error Status Register (LTESR)

The LBC has five registers for error management.

- The transfer error status register (LTESR) indicates the cause of an error.
- The transfer error check disable register (LTEDR) is used to enable (and disable) error checking.
- The transfer error check interrupt register (LTEIR) enables reporting of errors through an interrupt.
- The transfer error attributes register (LTEATR) captures source attributes of an error.
- The transfer error address register (LTEAR) captures the address of a transaction that caused an error.

LTESR, shown in Figure 12-13, is a write-one-to-clear register. Reading LTESR occurs normally; however, write operations can clear but not set bits. A bit is cleared whenever the register is written and the data in the corresponding bit location is a 1. For example, to clear only the write protect error bit (LTESR[WP]) without affecting other LTESR bits, 0x0400_0000 should be written to the register.

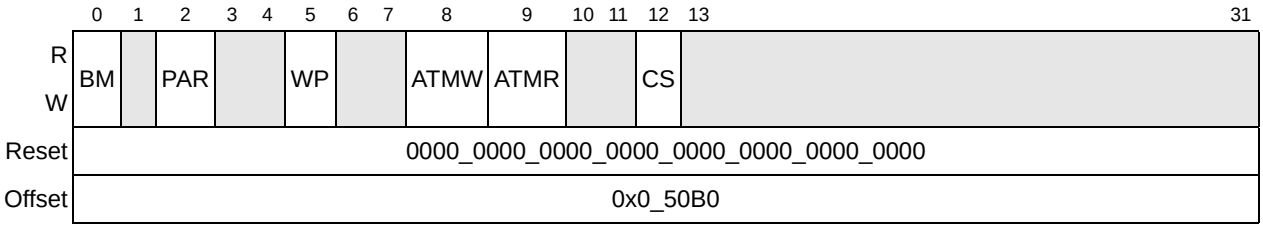


Figure 12-13. Transfer Error Status Register (LTESR)

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] × 8 bus clock cycles from the start of a transaction.
1	—	Reserved
2	PAR	Parity 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

Table 12-16. LTESR Field Descriptions (continued)

Bits	Name	Description
5	WP	Write protect error 0 No write protect error occurred. 1 A write was attempted to a local bus memory region that was defined as read-only in the memory controller. Usually, in this case, a bus monitor time-out will occur (as the cycle is not automatically terminated).
6–7	—	Reserved
8	ATMW	Atomic error write 0 No atomic write error occurred. 1 The subsequent write (WARA) to a memory bank did not occur within 256 bus clock cycles.
9	ATMR	Atomic error read 0 No atomic read error occurred. 1 The subsequent read (RAWA) to a memory bank did not occur within 256 bus clock cycles.
10–11	—	Reserved
12	CS	Chip select error 0 No chip select error occurred. 1 A transaction was sent to the LBC that did not hit any memory bank.
13–31	—	Reserved

12.3.1.11 Transfer Error Check Disable Register (LTEDR)

The transfer error check disable register (LTEDR), shown in Figure 12-14, is used to disable error checking. Note that control of error checking is independent of control of reporting of errors (LTEIR) through the interrupt mechanism.

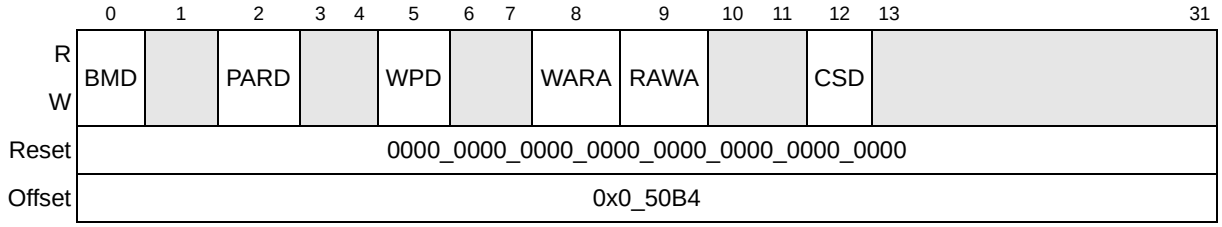


Figure 12-14. Transfer Error Check Disable Register (LTEDR)

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

Table 12-17. LTEDR Field Descriptions (continued)

Bits	Name	Description
2	PARD	Parity error checking disabled. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, PARD must be cleared and LTEIR[PARI] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 0 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)

The transfer error interrupt enable register (LTEIR), shown in Figure 12-15, is used to send or block error reporting through the LBC internal interrupt mechanism. Software should clear pending errors in LTESR before enabling interrupts. After an interrupt has occurred, clearing relevant LTESR error bits negates the interrupt.

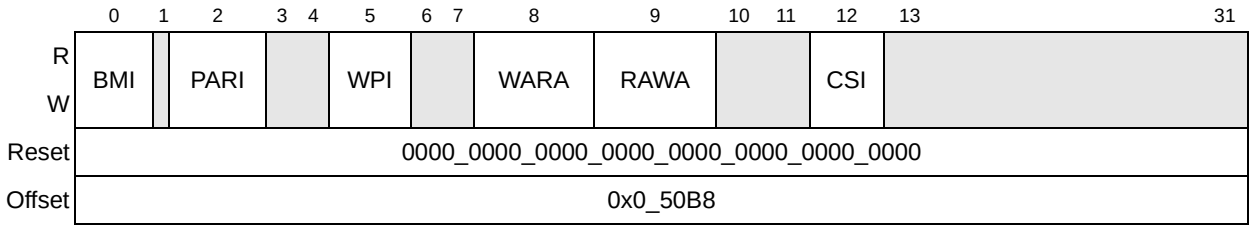


Figure 12-15. Transfer Error Interrupt Enable Register (LTEIR)

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. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, LTEDR[PARD] must be cleared and PARI must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 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
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. After LTEATR[V] has been set, software must clear this bit to allow LTEATR and LTEAR to update following any subsequent errors.

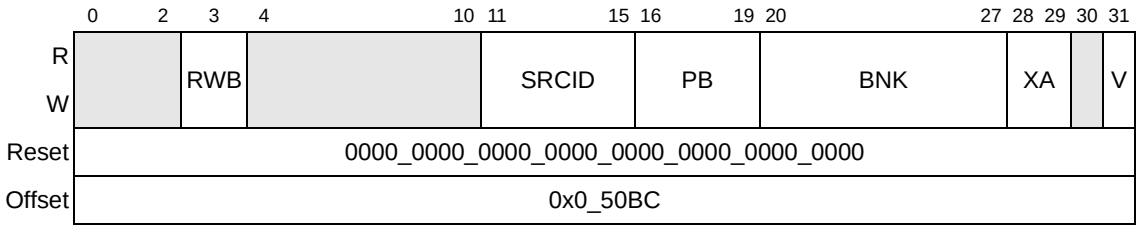


Figure 12-16. Transfer Error Attributes Register (LTEATR)

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	Captures 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 msbs of the 34-bit address of the transaction resulting in an error.
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)

The transfer error address register (LTEAR) is shown in Figure 12-17.

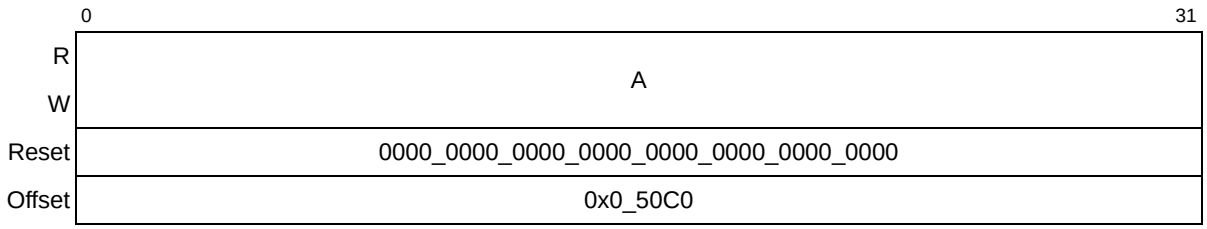


Figure 12-17. Transfer Error Address Register (LTEAR)

Table 12-20 describes LTEAR[A].

Table 12-20. LTEAR Field Descriptions

Bits	Name	Description
0–31	A	Transaction address for the error. Holds the 32 lsbs of the 34-bit address of the transaction resulting in an error.

12.3.1.15 Local Bus Configuration Register (LBCR)

The local bus configuration register (LBCR) is shown in Figure 12-18.

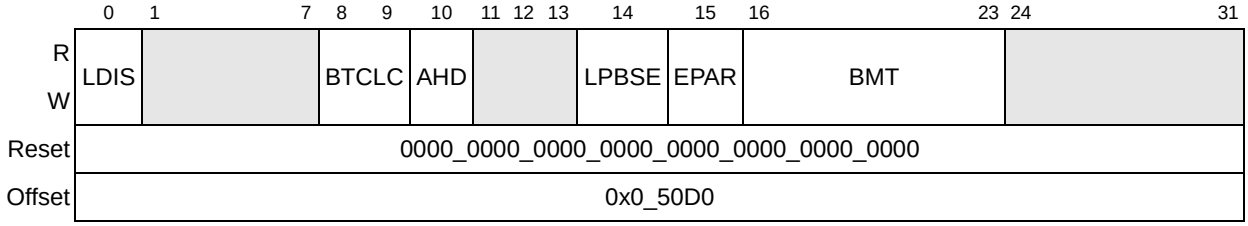


Figure 12-18. Local Bus Configuration Register

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	AHD	Address hold disable. Removes part of the hold time for LAD with respect to LALE in order to lengthen the LALE pulse 0 During address phases on the local bus, the LALE signal negates one platform clock period prior to the address being invalidated. At 333 MHz, this provides 3 ns of additional address hold time at the external address latch. 1 During address phases on the local bus, the LALE signal negates half of one platform clock period prior to the address being invalidated. This halves the address hold time, but extends the latch enable duration. This may be necessary for very high frequency designs.
11–13	—	Reserved
14	LPBSE	Enables parity byte select on $\overline{LGTA}$/LGPL4/LUPWAIT/LPBSE pin. 0 Parity byte select is disabled. $\overline{LGTA}$/LGPL4/LUPWAIT/LPBSE pin is available for memory control as LGPL4 (output) or $\overline{LGTA}$/LUP$\overline{WAIT}$ (input). 1 Parity byte select is enabled. $\overline{LGTA}$/LGPL4/LUPWAIT/LPBSE pin is dedicated as the parity byte select output, and $\overline{LGTA}$/LUPWAIT 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

Table 12-21. LBCR Field Descriptions (continued)

Bits	Name	Description
16–23	BMT	Bus monitor timing. Defines the bus monitor time-out period. Clearing BMT (reset value) selects the maximum count of 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 (CCB) clock to LBC bus frequency ratio. It also provides configuration bits for extra delay cycles for address and control signals.

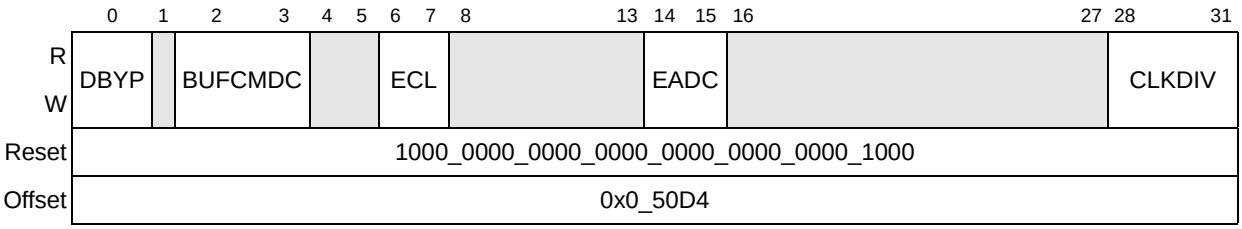


Figure 12-19. Clock Ratio Register (LCRR)

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

Table 12-22. LCRR Field Descriptions (continued)

Bits	Name	Description
14–15	EADC	External address delay cycles. Defines the number of cycles for the assertion of LALE. Note that LALE negates prior to the end of the final local bus clock, as controlled by LBCR[AHD]. 00 4 01 1 10 2 11 3
16–27	—	Reserved
28–31	CLKDIV	System (CCB) clock divider. Sets the frequency ratio between the system (CCB) clock and the memory bus clock. Only the values shown in the table below are allowed. 0000–0001 Reserved 0010 2 0011 Reserved 0100 4 0101–0111 Reserved 1000 8 1001–1111 Reserved

12.4 Functional Description

The LBC allows the implementation of memory systems with very specific timing requirements.

- The SDRAM machine provides an interface to SDRAMs using bank interleaving and back-to-back page mode to achieve high performance through a multiplexed address/data bus. An internal 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. It 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.

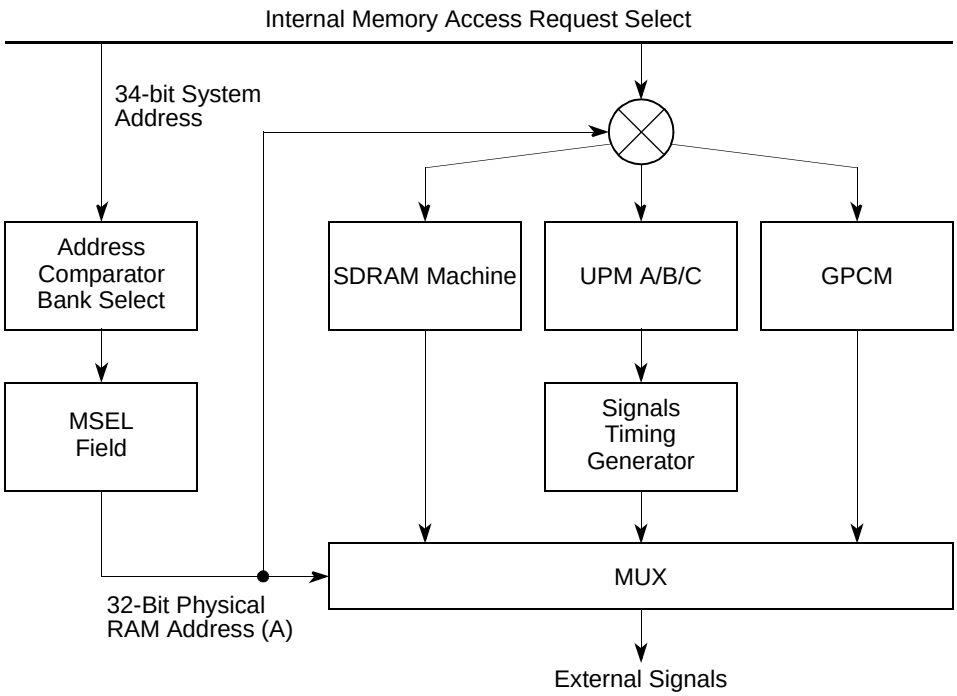


Figure 12-20. Basic Operation of Memory Controllers in the LBC

Each memory bank (chip select) can be assigned to any one of these three type of machines through the machine select bits of the base register for that bank (BR*n*[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

The following sections describe the basic architecture of the LBC.

12.4.1.1 Address and Address Space Checking

The defined base addresses are written to the BR*n* registers, while the corresponding address masks are written to the OR*n* registers. Each time a local bus access is requested, the internal transaction address is compared with each bank. Addresses are decoded by comparing the 19 msbs of the address, masked by OR*n*[XAM] and OR*n*[AM], with the base address for each bank (BR*n*[XBA] and BR*n*[BA]). If a match is found on a memory controller bank, the attributes defined in the BR*n* and OR*n* for that bank are used to control the memory access. If a match is found in more than one bank, the lowest-numbered bank handles the memory access (that is, bank 0 has priority over bank 1).

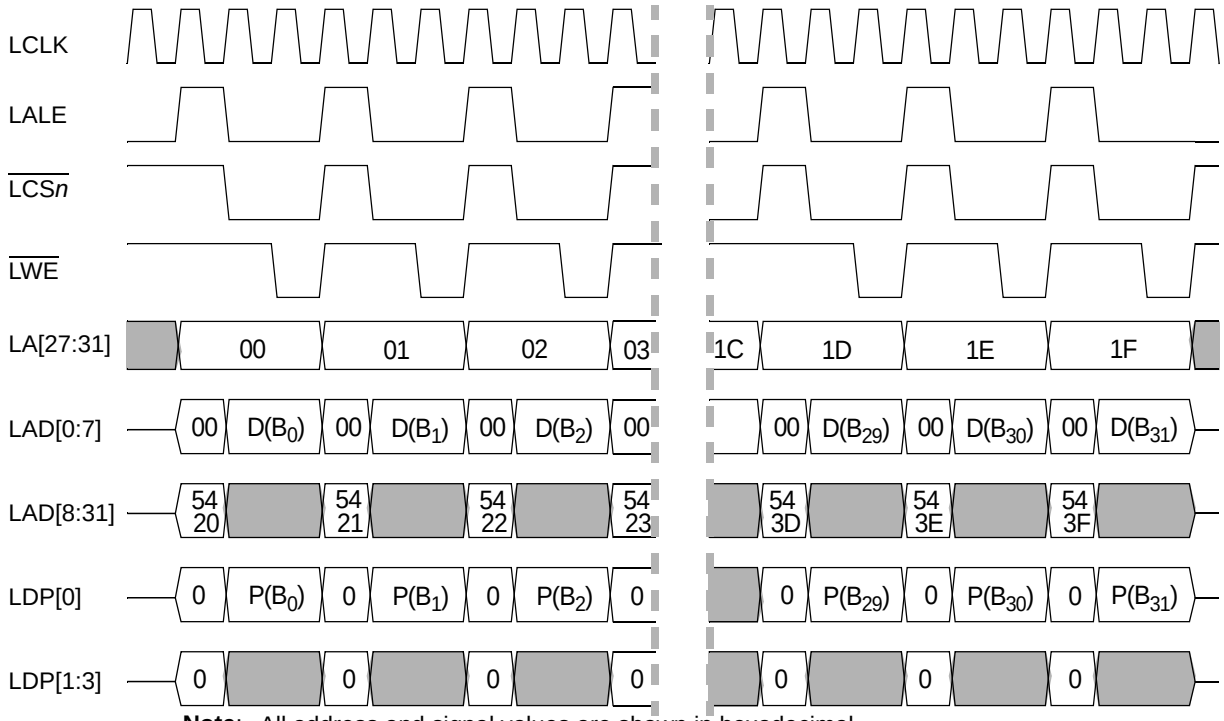
12.4.1.2 External Address Latch Enable Signal (LALE)

The local bus uses a multiplexed address/data bus. Therefore the LBC 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.

To ensure adequate hold time on the external address latch, LALE negates earlier than the address changes on LAD[0:31] during address phases. By default, LALE negates earlier by an amount equal to the platform clock period (which, divided by LCRR[CLKDIV], yields the bus clock). For example, if LBC is operating at 333 MHz internally, then an additional 3 ns of address hold time is introduced. However, when LCRR[CLKDIV] = 2 and the LCLK frequency exceeds 100 MHz, the duration of the shortened LALE pulse may not meet the minimum latch enable pulse width specifications of some latches. In such cases, setting LBCR[AHD] = 1 increases the LALE pulse width by half of one platform clock cycle, and decreases the address hold time by the same amount. At 333 MHz and with LCRR[CLKDIV] = 2, the duration of LALE would then be 4.5 ns, with 1.5 ns of hold time. If both longer hold time and longer LALE pulse duration are needed, then the address phase can be extended using the ORn[EAD] and LCRR[EADC] fields, and the LBCR[AHD] bit can be left at 0. However, this will add latency to all address tenures.

The frequency of LALE assertion varies across the three memory controllers. For GPCM, every assertion of $\overline{\text{LCS}n}$ is considered an independent access, and accordingly, LALE asserts prior to each such access. For example, GPCM driving an 8-bit port would assert LALE and $\overline{\text{LCS}n}$ 32 times in order to satisfy a 32-byte cache line transfer. 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 through 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 the LBC is generating its own burst address sequence.

To illustrate how a large transaction is handled by the LBC, Figure 12-21 shows LBC signals for GPCM performing a 32-byte write starting at address 0x5420. Note that during each of the 32 assertions of LALE, LA[27:31] exactly mirror LAD[27:31], but during data phases, only LAD[0:7] and LDP[0] are driven with valid data and parity, respectively.



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

Figure 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 the LBC asserts TA internally. In 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; a UPM generates TA only when a UPM pattern has the UTA RAM word bit set. Figure 12-22 shows LALE, TA (internal), and $\overline{\text{LCS}n}$. Note that TA and LALE are never asserted together, and that for the duration of LALE, $\overline{\text{LCS}n}$ (or any other control signal) remains negated or frozen.

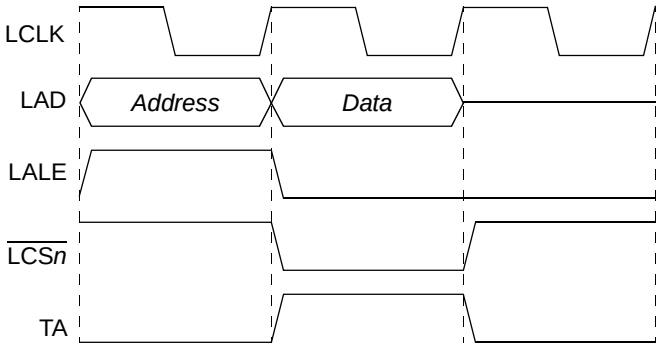


Figure 12-22. Basic LBC Bus Cycle with LALE, TA, and $\overline{\text{LCS}n}$

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

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

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*n*[ATOM]):

- Read-after-write atomic (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 atomic (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 are otherwise processed normally and do not change the reservation status. If the device fails to release the reservation within 256 bus clock cycles, the reservation is released and an atomic error is reported (if enabled).

12.4.1.6 Parity Generation and Checking (LDP)

Parity can be configured for any bank by programming BR*n*[DECC]. Parity is generated and checked on a per-byte basis using LDP[0:3] for the bank if BR*n*[DECC] = 01 (normal parity) or BR*n*[DECC] = 10 for read-modify-write (RMW) parity. Byte lane parity on LDP[0:3] is generated regardless of the BR*n*[DECC] setting. Note that RMW parity can be used only for 32-bit port size banks. LBCR[EPAR] determines the global type of parity (odd or even).

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 countdown until the data tenure completes and then idles if there is no pending transaction. Setting LTEDR[BMD] disables bus monitor error reporting through LTESR[BM]; however, 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*n* and OR*n*.

Figure 12-23 shows a simple connection between an 8-bit port size SRAM device and the LBC 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.

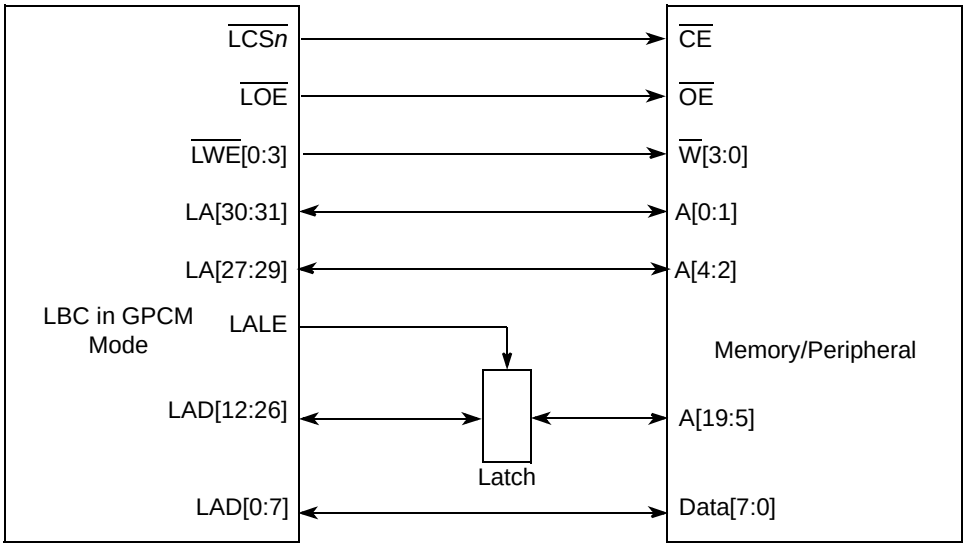


Figure 12-23. Local Bus to GPCM Device Interface

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 OR*n*[ACS] to specify $\overline{\text{LCS}}$ to meet this requirement.

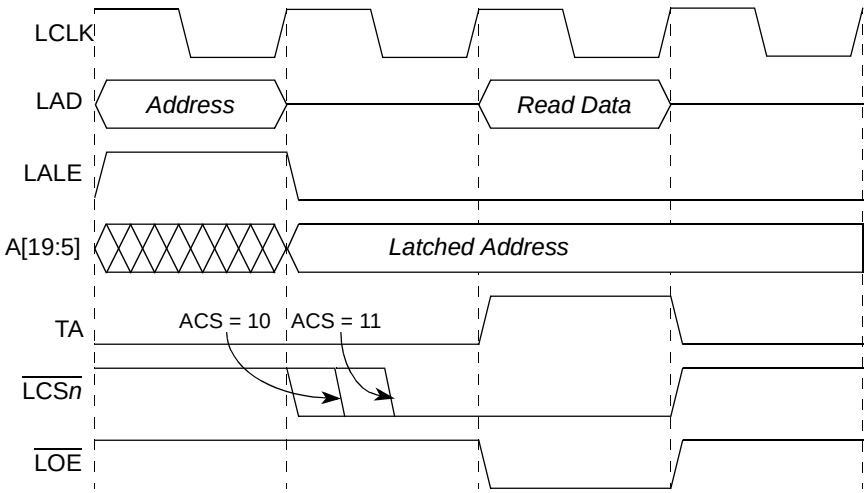


Figure 12-24. GPCM Basic Read Timing (XACS = 0, ACS = 1x, TRLX = 0, CLKDIV = 4,8)

12.4.2.1 Timing Configuration

If BR*n*[MSEL] selects the GPCM, the attributes for the memory cycle are taken from OR*n*. These attributes include the CSNT, ACS, XACS, SCY, TRLX, EHTR, and SETA fields. Table 12-23

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 LCS*n* Asserted	LCS*n* Negated to Address Change	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 (OR*n*[EAD] = 0; OR*n*[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 $\overline{\text{LCS}n}$ Asserted	$\overline{\text{LCS}n}$ 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 (OR*n*[EAD] = 0; OR*n*[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 $\overline{\text{LCS}n}$ Asserted	$\overline{\text{LCS}n}$ 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 (OR*n*[EAD]=0; OR*n*[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 $\overline{\text{LCS}n}$ Asserted	$\overline{\text{LCS}n}$ 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 1 cycle only (OR*n*[EAD]=0; OR*n*[EAD]=1 and LCRR[EADC]=01). Asserting LALE for more than 1 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 $\overline{\text{LCS}n}$ signal with different timings (with respect to the external address/data bus). $\overline{\text{LCS}n}$ can be driven in any of the following ways:

- Simultaneous with the latched memory address. (This refers to the externally latched address and not the address timing on LAD[0:31]. That is, chip select does not assert during LALE).
- One quarter of a clock cycle later (for LCRR[CLKDIV] = 4 or 8).
- One half of a clock cycle later (for LCRR[CLKDIV] = 2, 4, or 8).
- One clock cycle later (for LCRR[CLKDIV] = 4), when OR*n*[XACS] = 1.
- Two clock cycles later (for LCRR[CLKDIV] = 2, 4, or 8), when OR*n*[XACS] = 1.
- Three clock cycles later (for LCRR[CLKDIV] = 2, 4, or 8), when OR*n*[XACS] = 1 and OR*n*[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, $\overline{\text{LCS}n}$ asserts identically for OR*n*[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 OR*n*[SCY] and OR*n*[TRLX]. Internal generation of transfer acknowledge is enabled if OR*n*[SETA] = 0. If $\overline{\text{LGTA}}$ is asserted externally two bus clock cycles or more before the wait state counter has expired (to allow for synchronization latency), the current memory cycle is terminated by $\overline{\text{LGTA}}$; otherwise it is terminated by the expiration of the wait state counter. Regardless of the setting of OR*n*[SETA], wait states prolong the assertion duration of both $\overline{\text{LOE}}$ and $\overline{\text{LWE}n}$ in the same manner. When TRLX = 1, the number of wait states inserted by the memory controller is doubled from OR*n*[SCY] cycles to 2 × OR*n*[SCY] cycles, allowing a maximum of 30 wait states.

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, $\overline{\text{LCS}n}$ is connected directly to $\overline{\text{CE}}$ of the memory device. The $\overline{\text{LWE}}$[0:3] signals are connected to the respective $\overline{\text{WE}}$[3:0] signals on the memory device where each $\overline{\text{LWE}}$[0:3] signal corresponds to a different data byte.

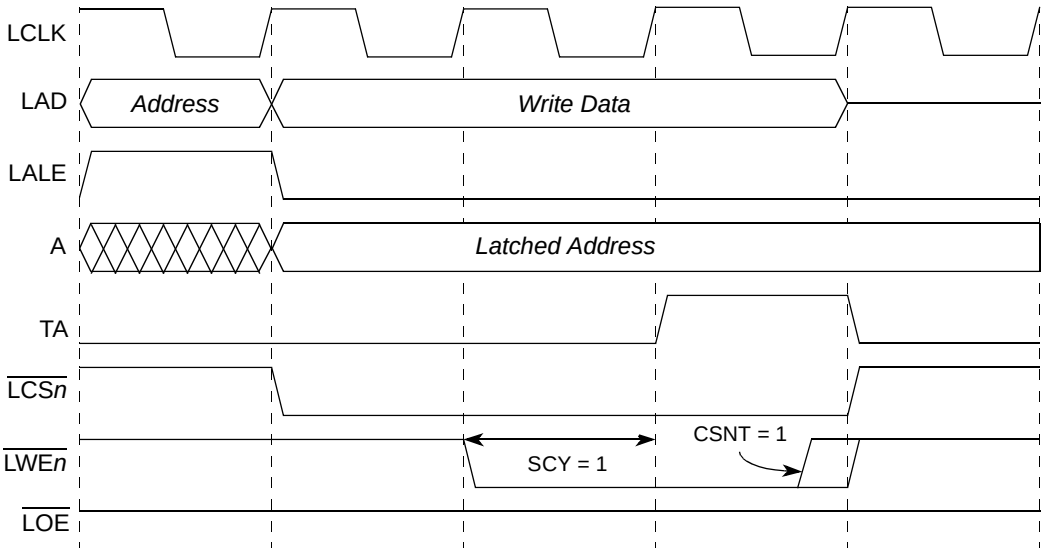


Figure 12-25. GPCM Basic Write Timing (XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0, CLKDIV = 4 or 8)

As Figure 12-25 shows, the timing for $\overline{\text{LCS}n}$ is the same as for the latched address. The strobes for the transaction are supplied by $\overline{\text{LOE}}$ or $\overline{\text{LWE}n}$, depending on the transaction direction—read or write (write case shown in Figure 12-25). OR*n*[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[CLKDIV] = 4 or 8. For example, when ACS = 00 and CSNT = 1, $\overline{\text{LWE}n}$ is negated one quarter of a clock earlier, as shown in Figure 12-25. If LCRR[CLKDIV] = 2, $\overline{\text{LWE}n}$ is negated either coincident with $\overline{\text{LCS}n}$ 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 ≠ 00).
- The number of wait states specified by SCY is doubled, providing up to 30 wait states.
- The extended hold time on read accesses (EHTR) is extended further.
- $\overline{\text{LCS}n}$ signals are negated 1 cycle earlier during writes (but only if ACS ≠ 00).
- $\overline{\text{LWE}}$[0:3] signals are negated 1 cycle earlier during writes.

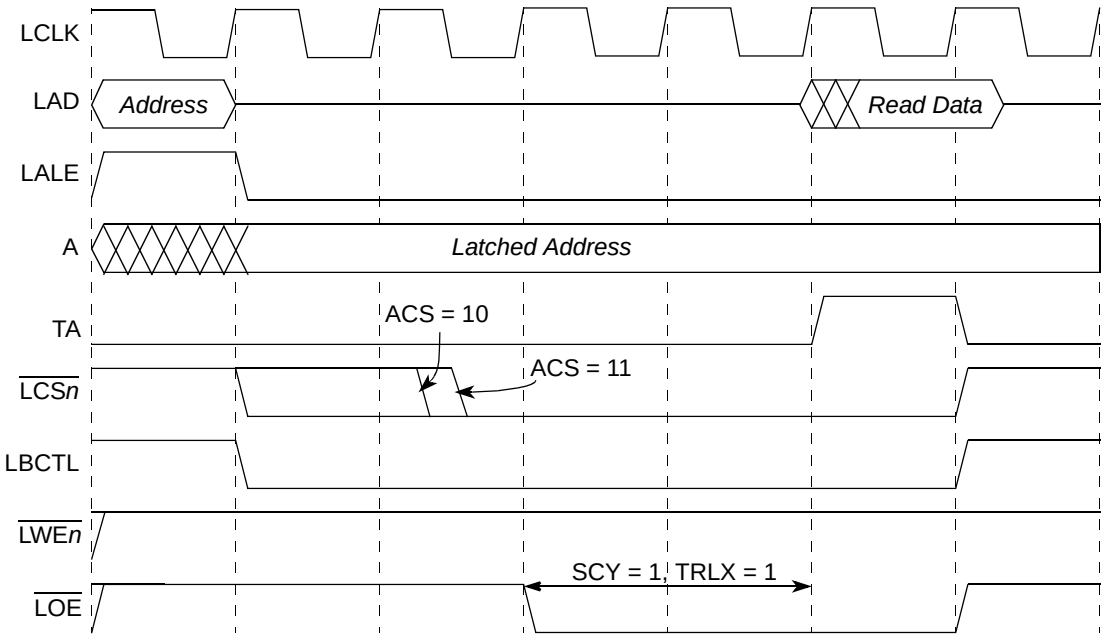


Figure 12-26. GPCM Relaxed Timing Read (XACS = 0, ACS = 1x, SCY = 1, CSNT = 0, TRLX = 1, CLKDIV = 4 or 8)

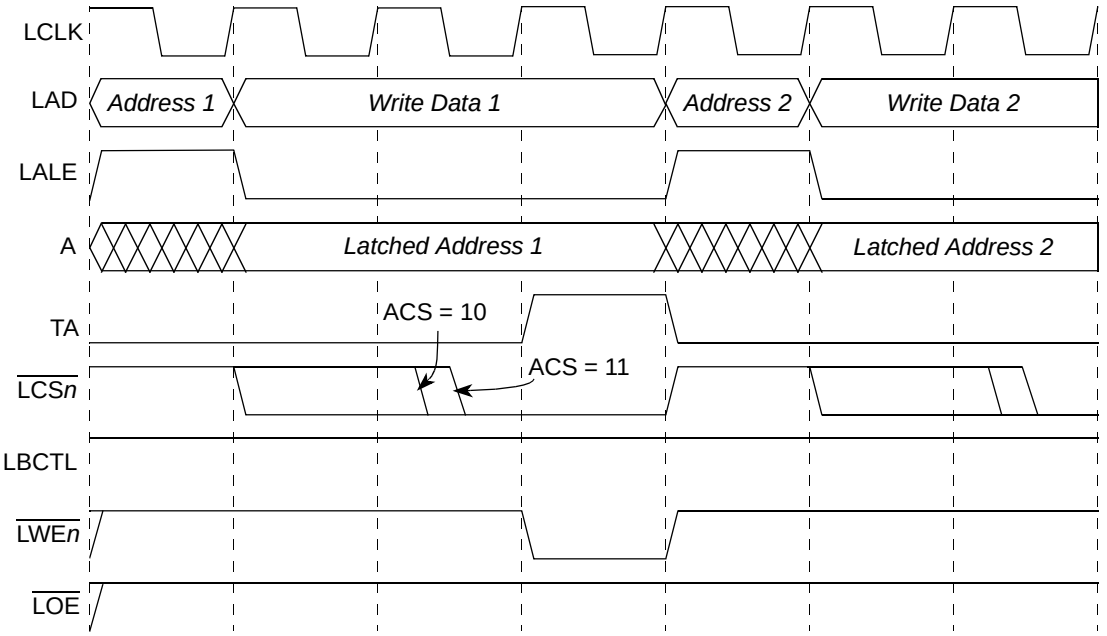


Figure 12-27. GPCM Relaxed Timing Back-to-Back Writes (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1, CLKDIV = 4 or 8)

Figure 12-26 and Figure 12-27 show relaxed timing read and write transactions. The effect of CLKDIV = 2 for these examples is only to delay the assertion of $\overline{\text{LCS}n}$ 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, $\overline{\text{LCS}n}$ is also negated one clock earlier.

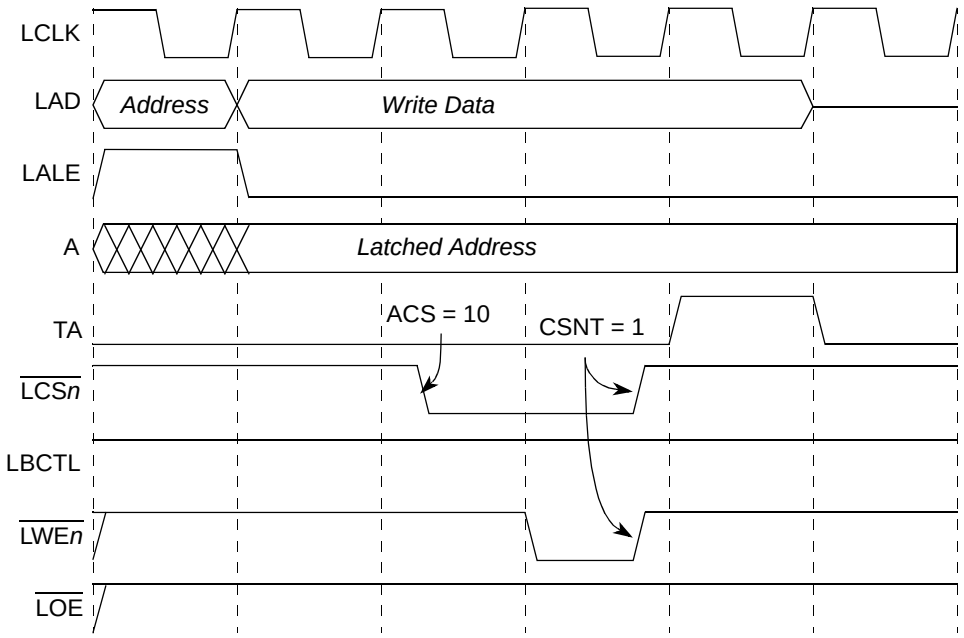


Figure 12-28. GPCM Relaxed Timing Write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1, CLKDIV = 4 or 8)

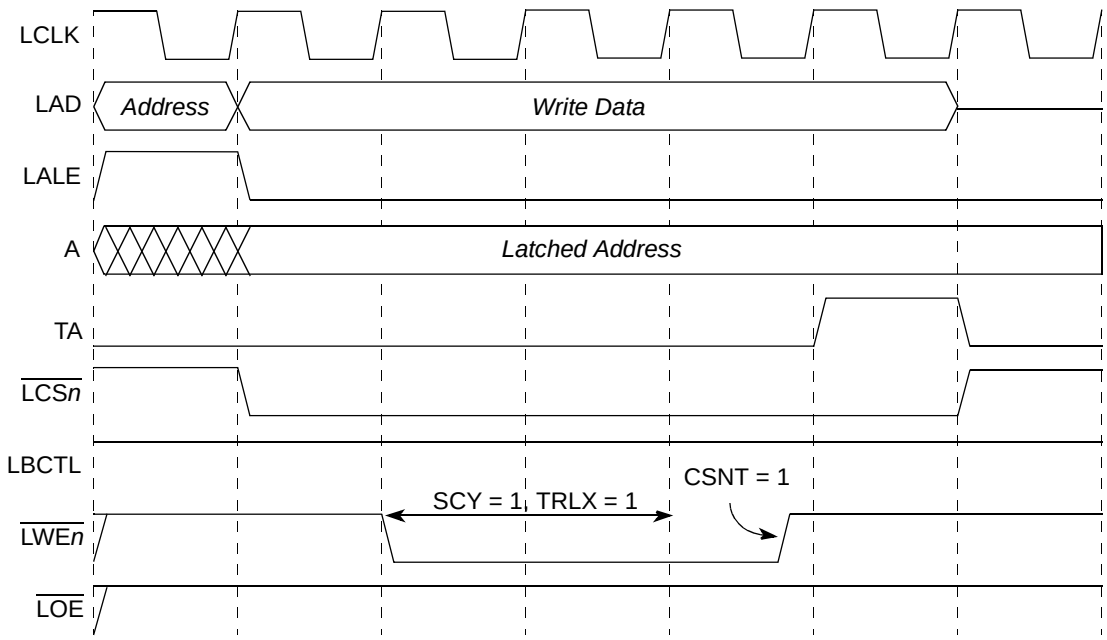


Figure 12-29. GPCM Relaxed Timing Write (XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1, CLKDIV = 4 or 8)

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

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*n*[TRLX,EHTR]. Any access following a read access to the slower memory bank is delayed by the number of clock cycles specified in Table 12-6 in addition to any existing bus turnaround cycle. The final bus turnaround cycle is automatically inserted by the LBC for reads, regardless of the setting of OR*n*[EHTR].

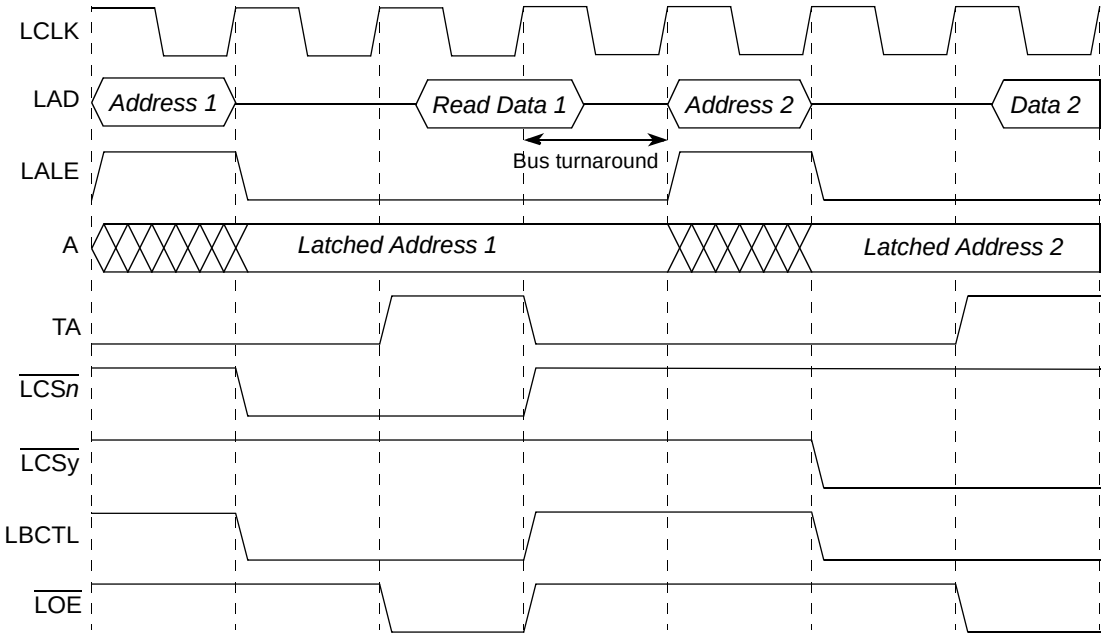


Figure 12-30. GPCM Read Followed by Read (TRLX = 0, EHTR = 0, Fastest Timing)

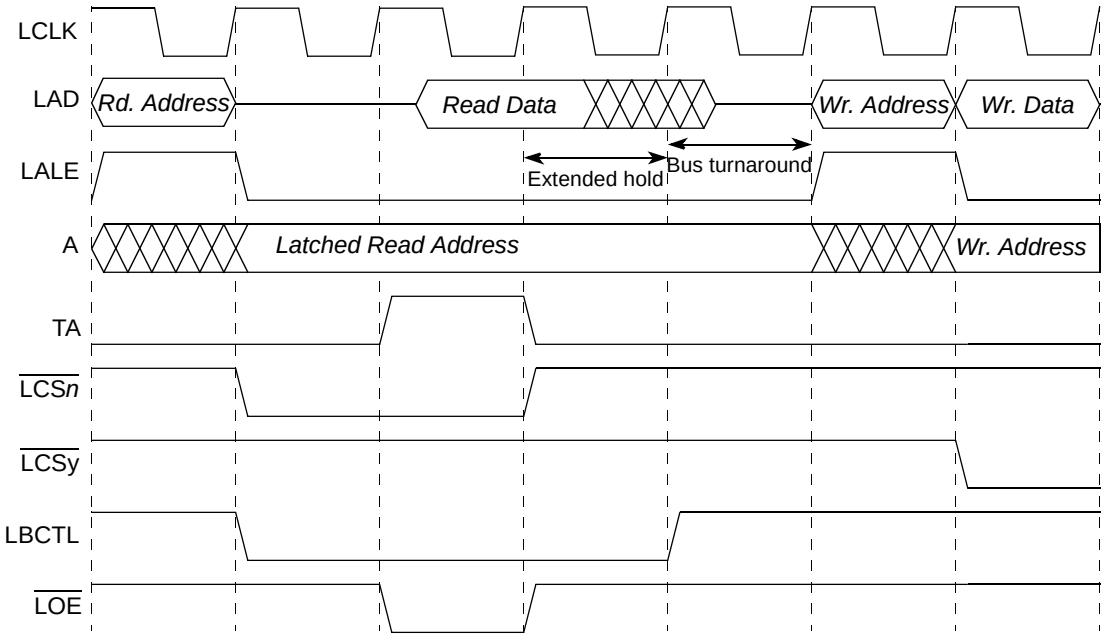


Figure 12-31. GPCM Read Followed by Write (TRLX = 0, EHTR = 1, 1-Cycle Extended Hold Time on Reads)

12.4.2.3 External Access Termination (LGTA)

External access termination is supported by the GPCM using the asynchronous LGTA input signal, which is synchronized and sampled internally by the local bus. If, during assertion of LCS*n*, the sampled LGTA signal is asserted, it is converted to an internal generation of transfer acknowledge, which terminates the current GPCM access (regardless of the setting of OR*n*[SETA]). LGTA should be asserted for at least one bus cycle to be effective. Note that because LGTA is synchronized, bus termination occurs two cycles after LGTA assertion, so in case of read cycle, the device still must drive data as long as LOE is asserted.

The user selects whether transfer acknowledge is generated internally or externally (LGTA) by programming OR*n*[SETA]. Asserting LGTA always terminates an access, even if OR*n*[SETA] = 0 (internal transfer acknowledge generation), but it is the only means by which an access can be terminated if OR*n*[SETA] = 1. The timing of LGTA is illustrated by the example in Figure 12-32.

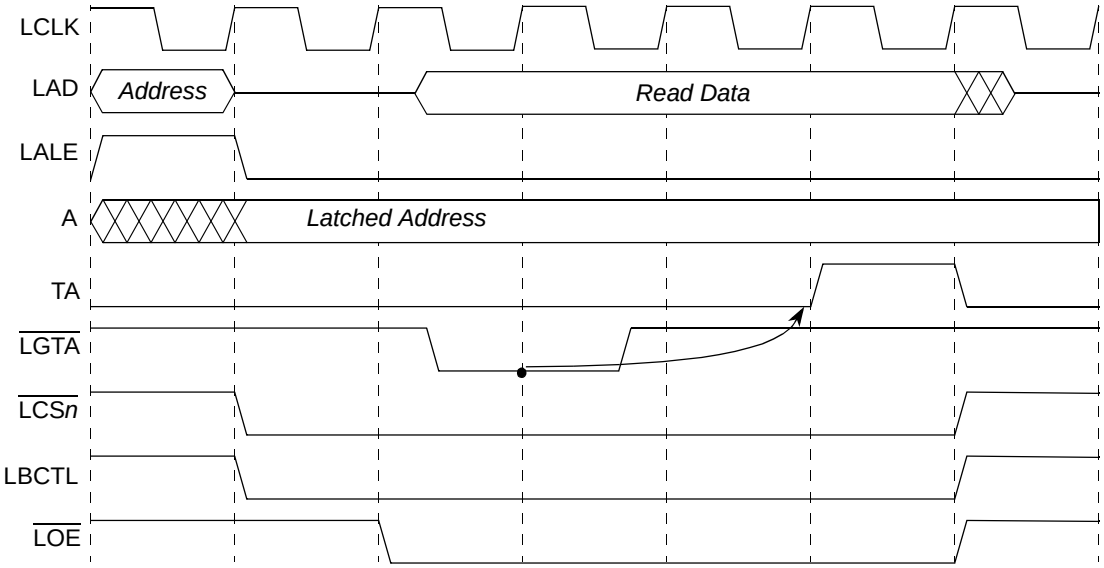


Figure 12-32. External Termination of GPCM Access

12.4.2.4 Boot Chip-Select Operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization. LCS0 is the boot chip-select output; its operation differs from other external chip-select outputs after a system reset. When the core begins accessing memory after system reset, 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. 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 XBA PS DECC WP MSEL ATOM V	0000_0000_0000_0000_0 00 From pin during reset. 00 0 000 00 1
OR0	AM XAM BCTLD CSNT ACS XACS SCY SETA TRLX EHTR EAD	0000_0000_0000_0000_0 00 0 1 11 1 1111 0 1 1 1

12.4.3 SDRAM Machine

The LBC 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 LBCs dedicated LSDA10 signal, while the remaining address bits (except A10) are latched from LAD[20:26].

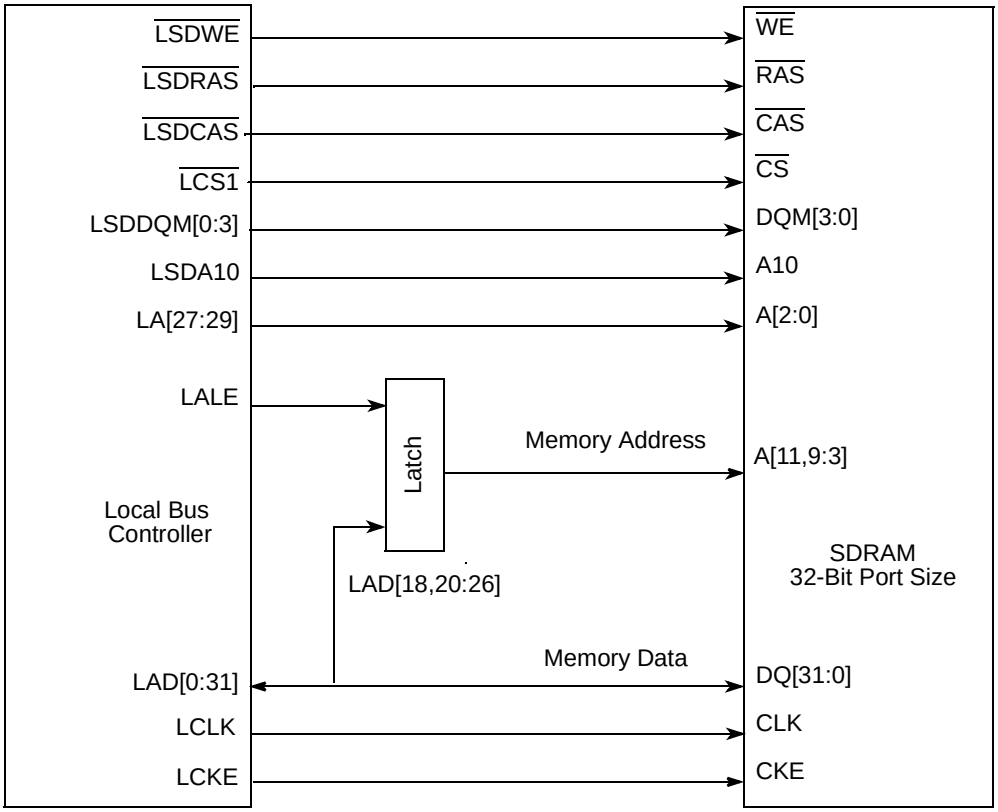


Figure 12-33. Connection to a 32-Bit SDRAM with 12 Address Lines

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*n*, BR*n*, 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 port size. 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 the LBC is negated. 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 the 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 OR*n* 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 OR*n*[PMSEL] = 1.

12.4.3.5 Page Management

The LBC can manage at most four 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 that 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 OR*n*[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 address bus bits 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], supporting page-based interleaving. The lsb of the logical row address (A*n* in Figure 12-34) is aligned with the connected lsb of LAD (bits 29, 30, and 31 for port sizes of 32, 16, and 8 bits, respectively).

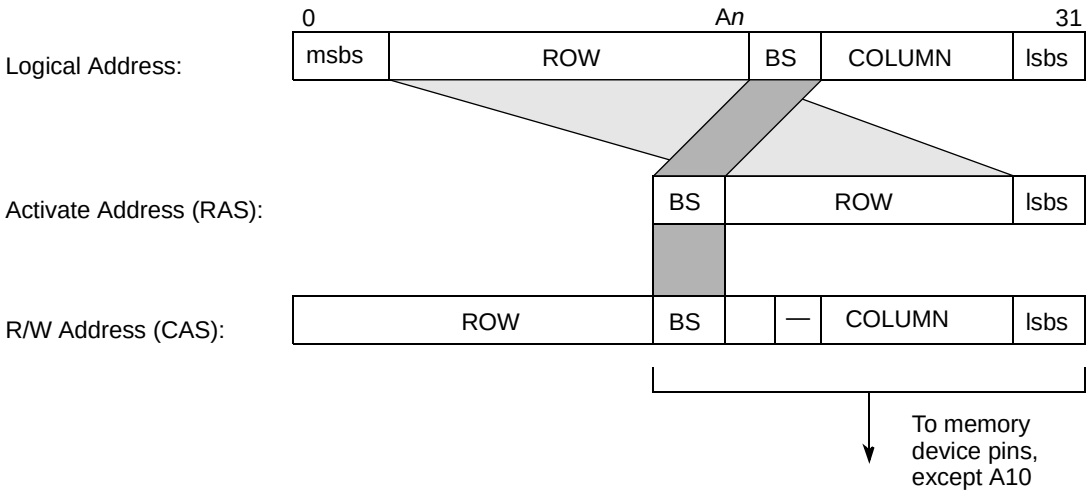


Figure 12-34. SDRAM Address Multiplexing

Note that during normal operation (read/write), a full 32-bit address that includes row and column is generated on LAD[0:31]. 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*n* 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

The precharge-to-activate interval parameter, controlled by LSDMR[PRETOACT]. defines the earliest timing for an ACTIVATE or REFRESH command after a PRECHARGE command to the same SDRAM bank.

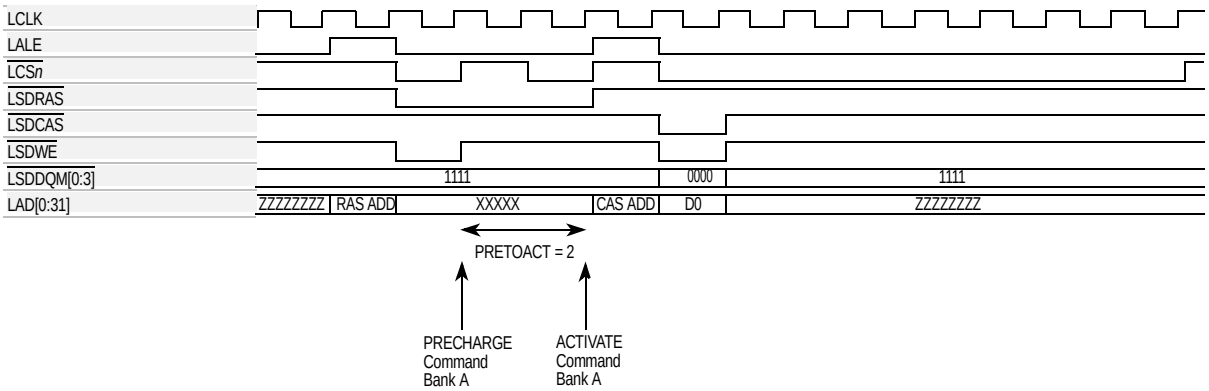


Figure 12-35. PRETOACT = 2 (2 Clock Cycles)

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.

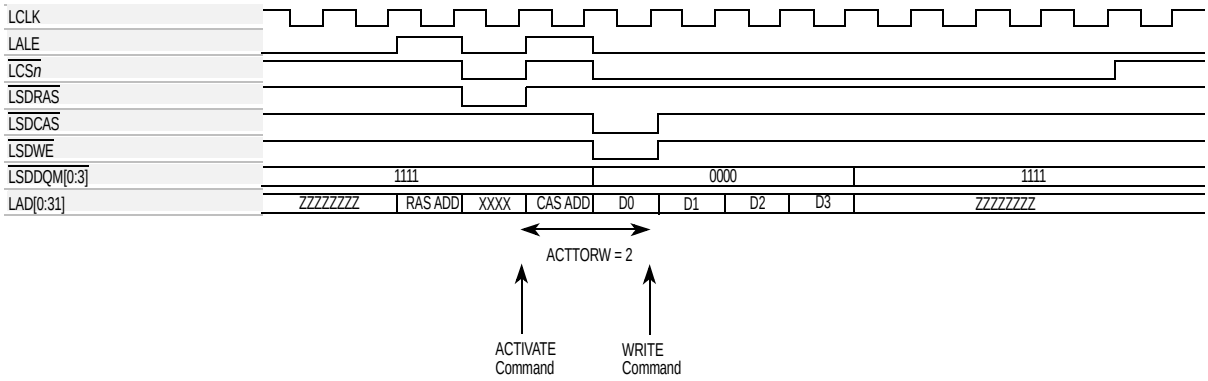


Figure 12-36. ACTTORW = 2 (2 Clock Cycles)

12.4.3.7.3 Column Address to First Data Out—$\overline{\text{CAS}}$ 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.

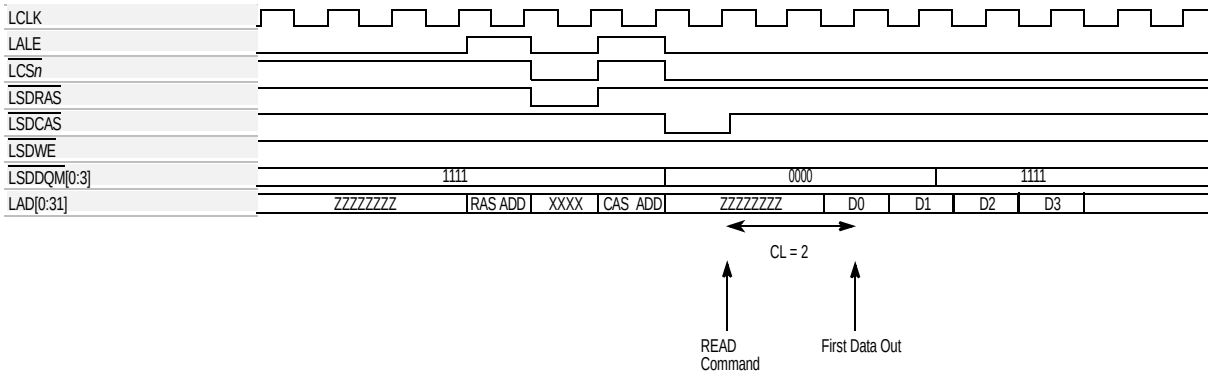


Figure 12-37. CL = 2 (2 Clock Cycles)

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.

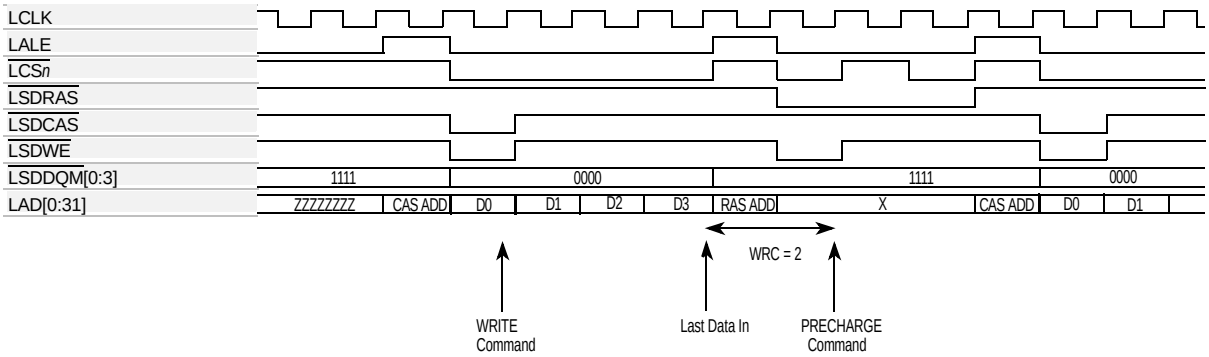


Figure 12-38. WRC = 2 (2 Clock Cycles)

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.

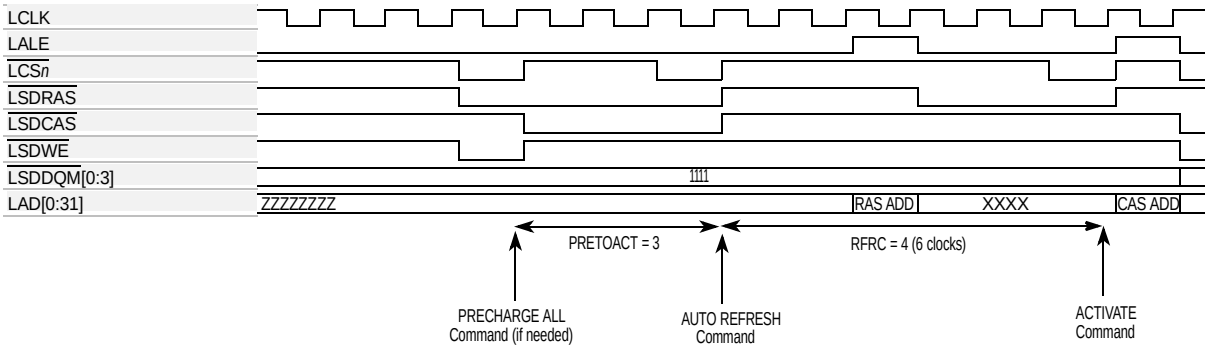


Figure 12-39. RFRC = 4 (6 Clock Cycles)

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.

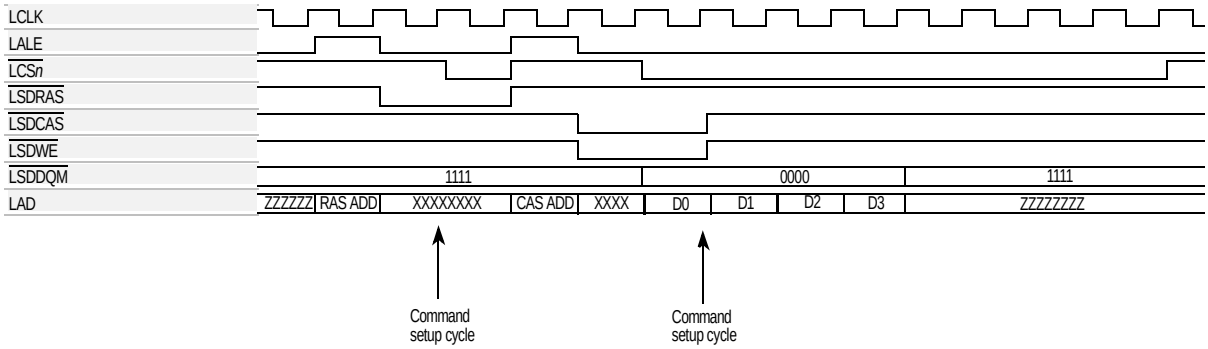


Figure 12-40. BUFCMD = 1, LCRR[BUFCMDC] = 2

12.4.3.8 SDRAM Interface Timing

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

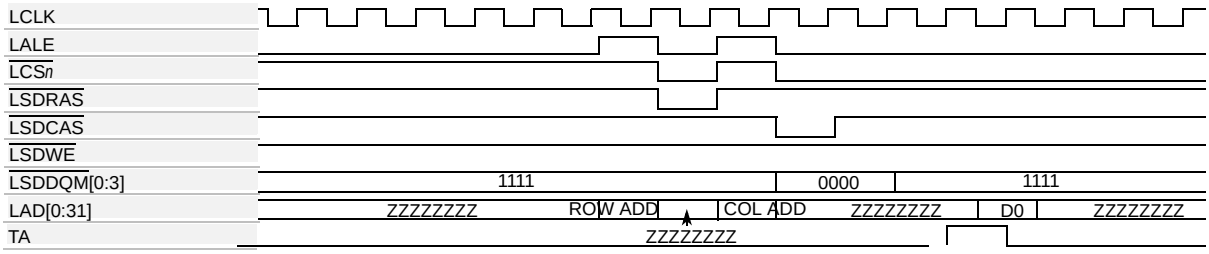


Figure 12-41. SDRAM Single-Beat Read, Page Closed, CL = 3

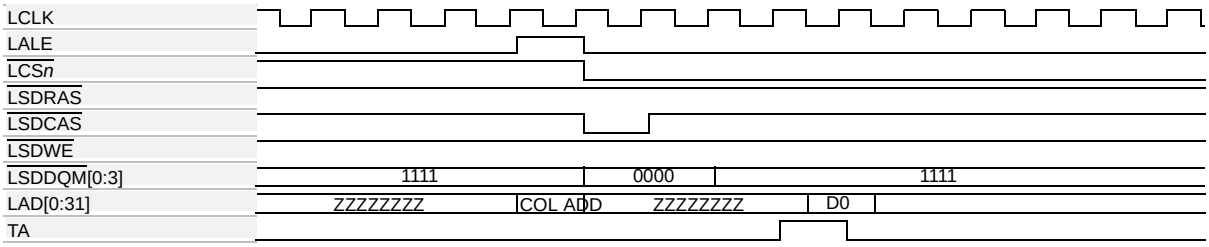


Figure 12-42. SDRAM Single-Beat Read, Page Hit, CL = 3

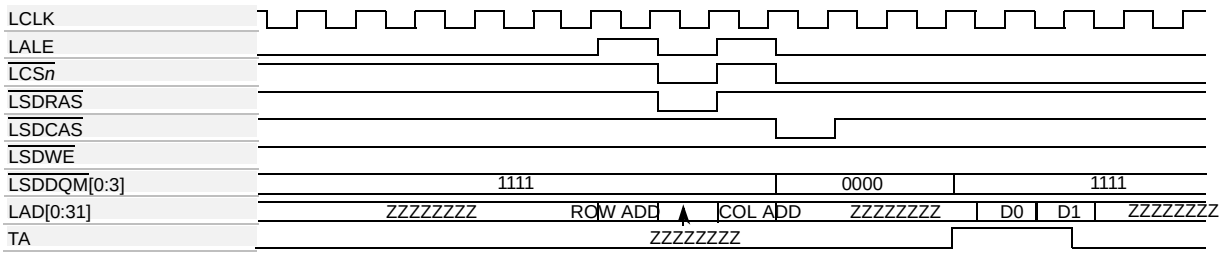


Figure 12-43. SDRAM Two-Beat Burst Read, Page Closed, CL = 3

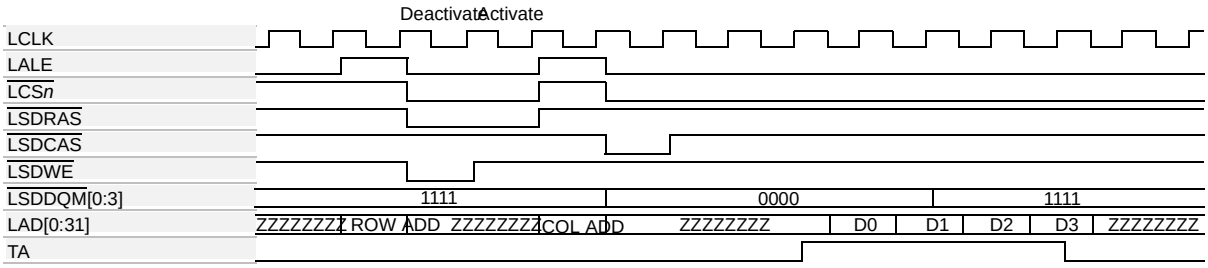


Figure 12-44. SDRAM Four-Beat Burst Read, Page Miss, CL = 3

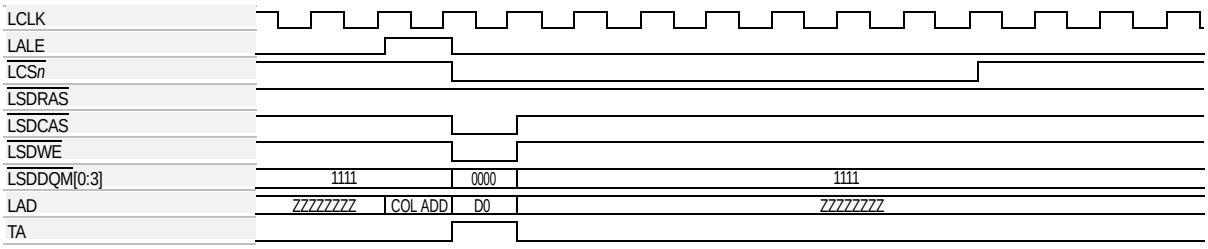


Figure 12-45. SDRAM Single-Beat Write, Page Hit.

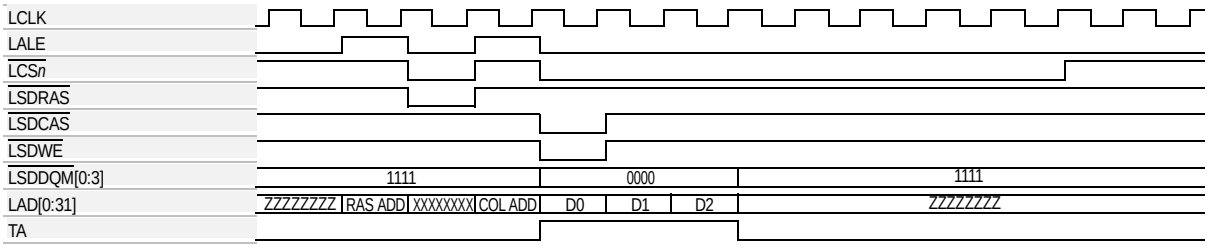


Figure 12-46. SDRAM Three-Beat Write, Page Closed

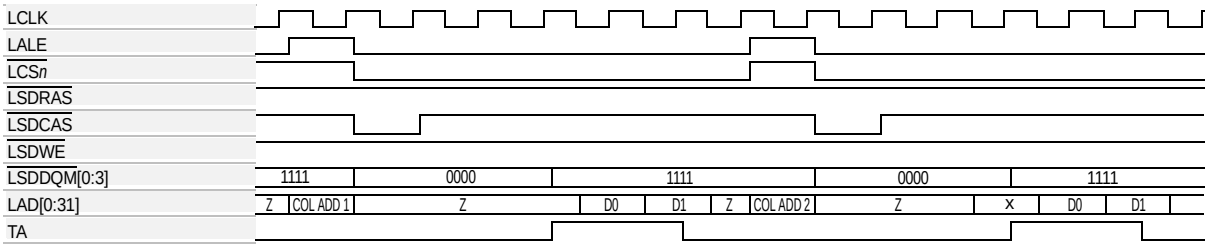


Figure 12-47. SDRAM Read-after-Read Pipelined, Page Hit, CL = 3

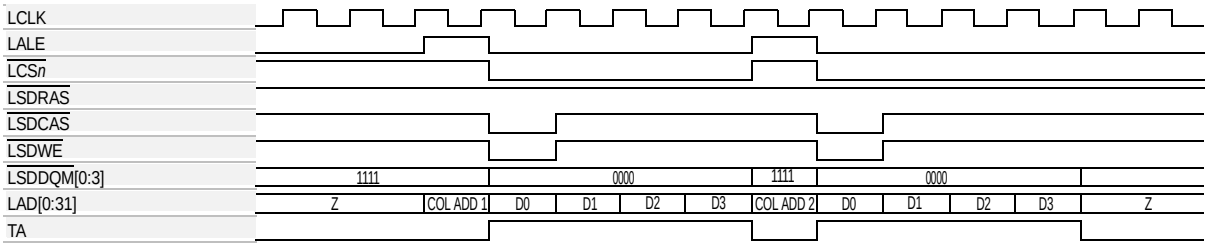


Figure 12-48. SDRAM Write-after-Write Pipelined, Page Hit

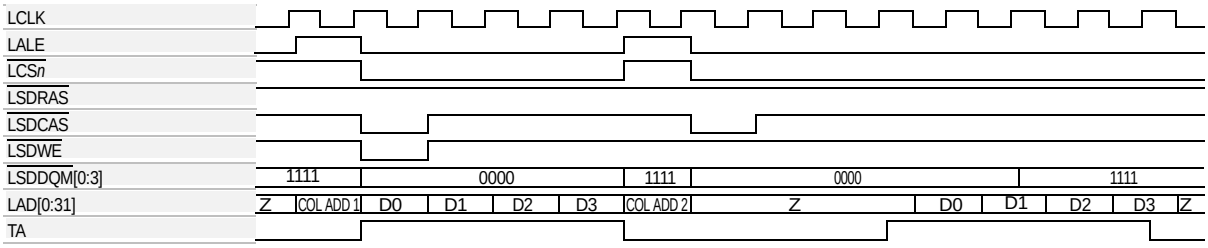


Figure 12-49. SDRAM Read-after-Write Pipelined, Page Hit

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.

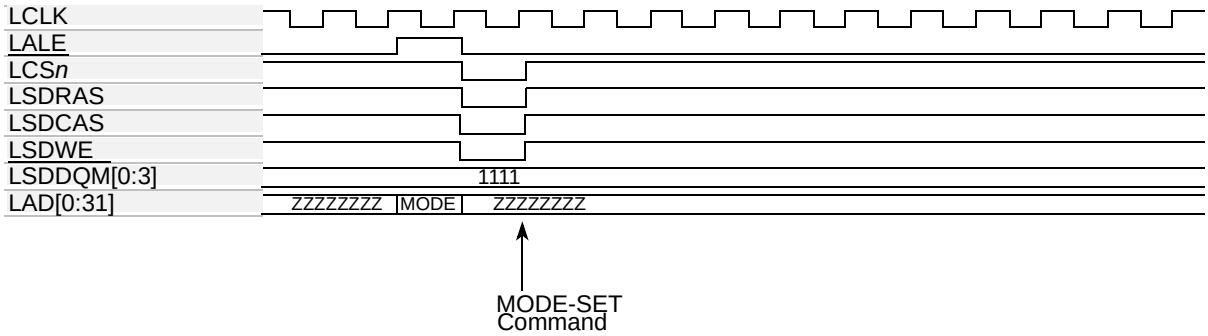


Figure 12-50. SDRAM MODE-SET Command

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].

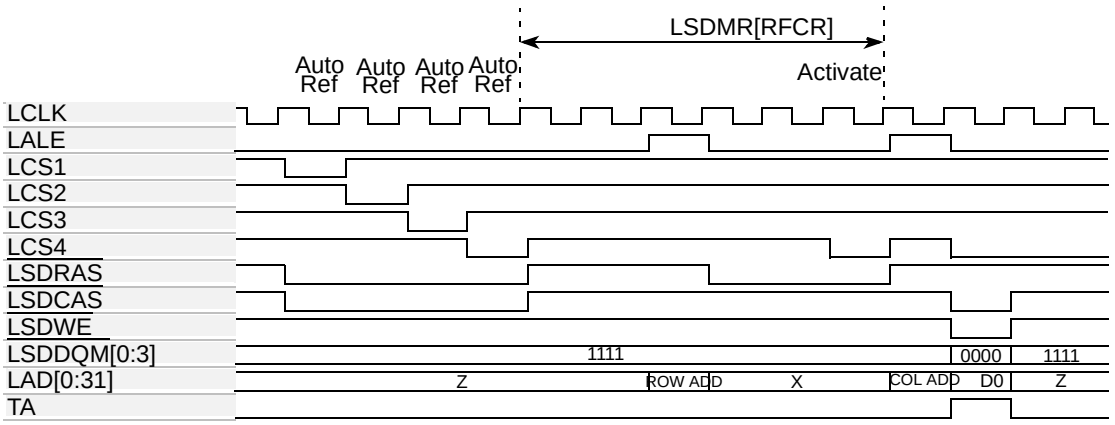


Figure 12-51. SDRAM Bank-Staggered Auto-Refresh Timing

12.4.4 User-Programmable Machines (UPMs)

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[1] ($\overline{\text{LCS}}n$, $\overline{\text{LBS}}$[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.

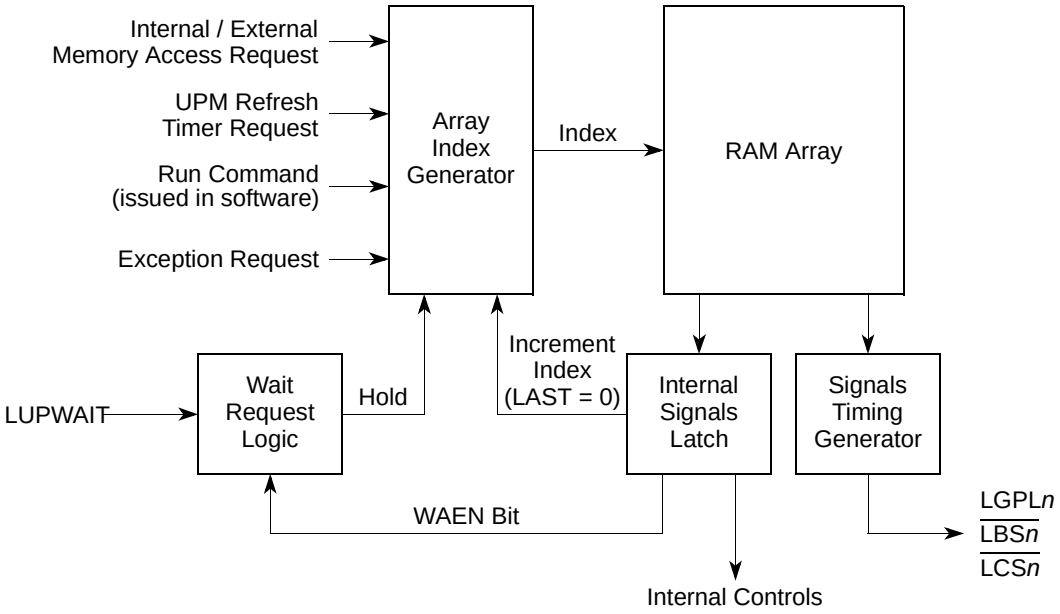


Figure 12-52. User-Programmable Machine Functional Block Diagram

1.If the LGPL4/$\overline{\text{LGTA}}$/LUPWAIT/LPBSE signal is used as both an input and an output, a weak pullup is required. Refer to the hardware specification for details regarding termination options.

The following events initiate a UPM cycle:

- Any internal device requests an external memory access to an address space mapped to a chip-select serviced by the UPM
- A UPM refresh timer expires and requests a transaction, such as a DRAM refresh
- A bus monitor time-out error during a normal UPM cycle redirects the UPM to execute an exception sequence

The RAM array contains 64 words of 32-bits each. The signal timing generator loads the RAM word from the RAM array to drive the general-purpose lines, byte-selects, and chip-selects. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller and the current request is frozen.

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.

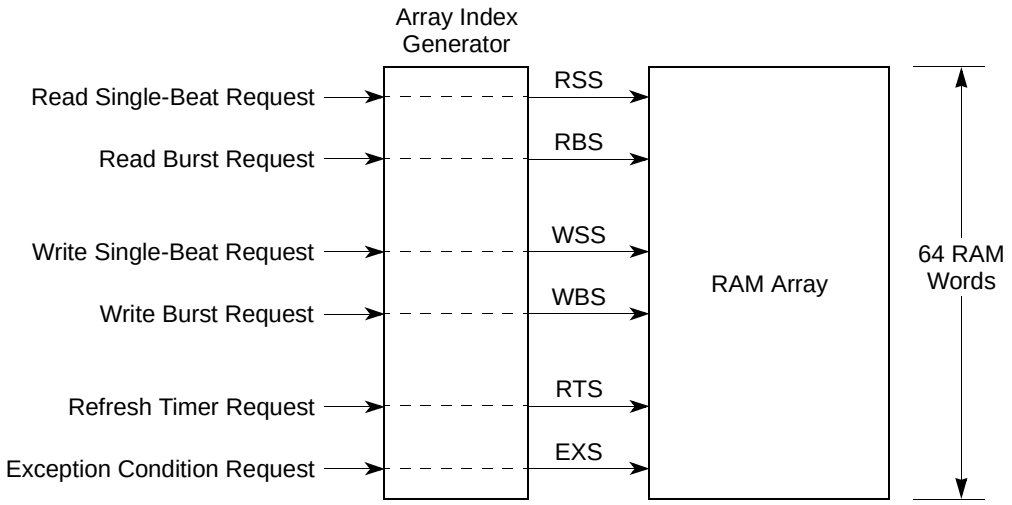


Figure 12-53. RAM Array Indexing

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 4 double words regardless of port size. For 32-bit accesses, the burst cycle starts with one transfer start but ends after eight transfer acknowledges, whereas an 8-bit device requires 32 transfer acknowledges.

The user must ensure that patterns for single-beat transfers contain one, and only one, transfer acknowledge (UTA bit in RAM word set high) and for a burst transfer, contain the exact number of transfer acknowledges required.

Any transfers that do not naturally fit single or burst transfers are synthesized as a series of single transfers. These accesses are treated by the UPM as back-to-back, single-beat transfers. Burst transfers can also be inhibited by setting OR*n*[BI]. Burst performance can be achieved by ensuring that UPM transactions are 32-byte aligned with a transaction size being some multiple of 32-bytes, which is a natural fit for cache-line transfers, for example.

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.

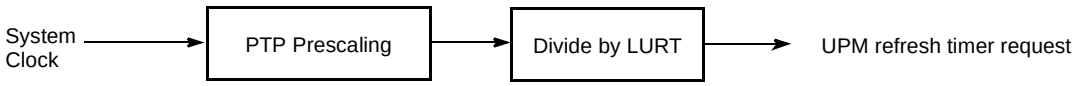


Figure 12-54. Memory Refresh Timer Request Block Diagram

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 M*x*MR[RFEN] bits are set.

Note that the UPM refresh timer request should 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 a RUN command is used to run the cycle. The UPM runs the pattern beginning at the specified RAM location until it encounters a RAM word with its LAST bit set. The RUN command is issued by setting M*x*MR[OP] = 11 and accessing UPM*n* memory region with any write transaction that hits the corresponding UPM machine. M*x*MR[MAD] determines the starting address in the RAM array for the pattern.

Note that transfer acknowledges (UTA bit in the RAM word) are ignored for software (RUN command) requests, and hence the LAD signals remain high-impedance unless the normal initial LALE occurs or the RUN pattern causes assertion of LALE to occur on changes to the RAM word AMX field.

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 device's timing requirements without losing data. The mechanism is the exception pattern that defines how the UPM negates its signals in a controlled manner.

12.4.4.2 Programming the UPMs

The UPM is a micro sequencer that requires microinstructions or RAM words to generate signal timings for different memory cycles. Follow these steps to program UPMs:

1. Set up BR*n* and OR*n* registers.
2. Write patterns into the RAM array.
3. Program MRTPR, LURT and MAMR[RFEN] if refresh is required.
4. Program M*x*MR.

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

Note that the UPM memory region must be cache-inhibited or write-through (the MMU page must have the I or W bit set) during the time that the UPM array is being written. If the memory is to be cacheable and/or copyback, the MMU must be set accordingly after the UPM array is initialized.

RAM array contents may also be read for debug purposes, for example, by alternating dummy read transactions, each time followed by reads of MDR (when M*x*MR[OP] = 10).

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

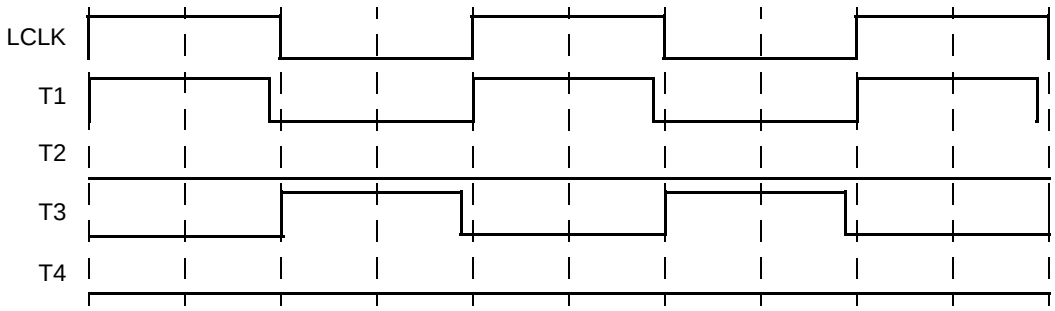


Figure 12-55. UPM Clock Scheme for LCRR[CLKDIV] = 2

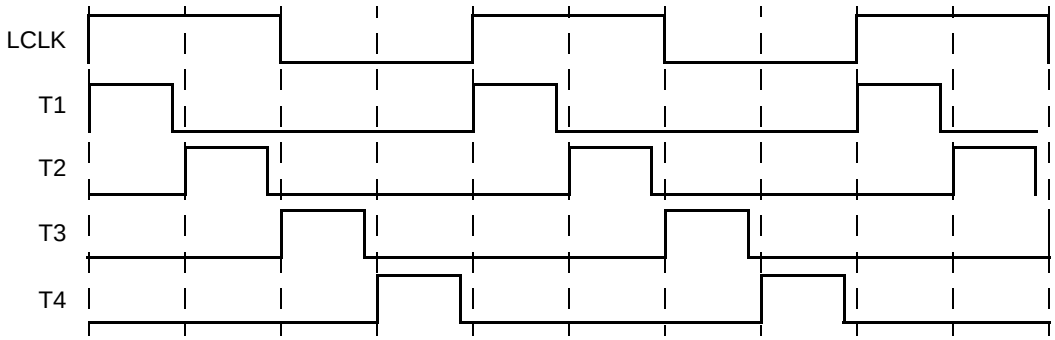


Figure 12-56. UPM Clock Scheme for LCRR[CLKDIV] = 4 or 8

12.4.4.4 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 $\overline{\text{LCS}n}$ is for the bank that matches the current address. The selected $\overline{\text{LBS}}$ is for the byte lanes read or written by the access.

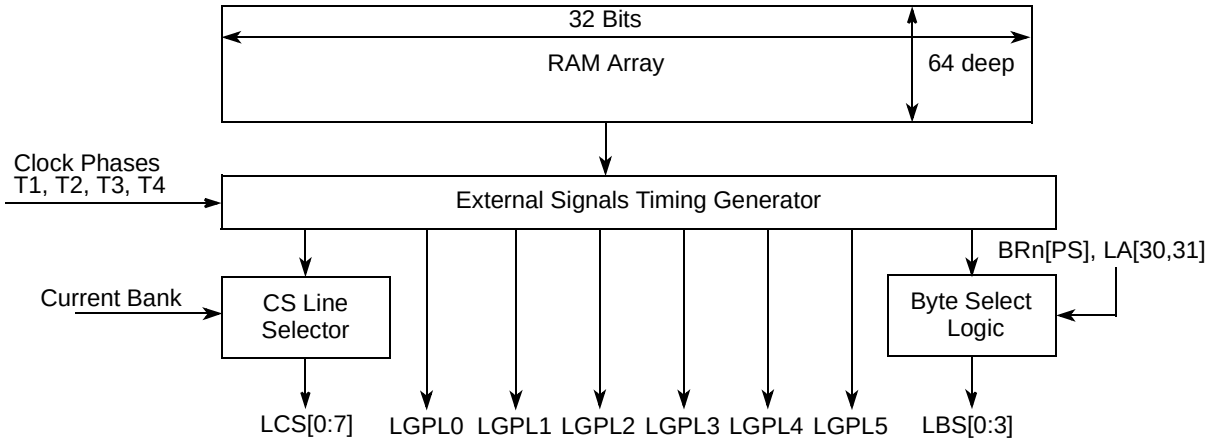


Figure 12-57. RAM Array and Signal Generation

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*n* and BST*n* bits determine the state of UPM signals $\overline{\text{LCS}n}$ and $\overline{\text{LBS}}$[0:3] at each quarter phase of the bus clock. 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 BST*n* 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 $\overline{\text{LCS}n}$ during bus clock quarter phase 1 if LCRR[CLKDIV] = 4 or 8. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ during bus clock half phase 1 if LCRR[CLKDIV] = 2.
1	CST2	Chip select timing 2. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ 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 $\overline{\text{LCS}n}$ during bus clock quarter phase 3 if LCRR[CLKDIV] = 4 or 8. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ during bus clock half phase 2 if LCRR[CLKDIV] = 2.
3	CST4	Chip select timing 4. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ during bus clock quarter phase 4.if LCRR[CLKDIV] = 4 or 8. Ignored when LCRR[CLKDIV] = 2.
4	BST1	Byte select timing 1. 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 BR*n*[PS] and the state of 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 BR*n*[PS] and the state of 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 BR*n*[PS] and the state of 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 BR*n*[PS] and the state of 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 M*x*MR[G0CL] 01 Reserved 10 0 11 1
10–11	G0H	General-purpose line 0 higher. Defines the state of LGPL0 during the bus clock quarter phases 3 and 4 (second half phase). 00 Value defined by M*x*MR[G0CL] 01 Reserved 10 0 11 1
12	G1T1	General-purpose line 1 timing 1. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 1 and 2 (first half phase).
13	G1T3	General-purpose line 1 timing 3. Defines the state (0 or 1) of LGPL1 during bus clock quarter phases 3 and 4 (second half phase)
14	G2T1	General-purpose line 2 timing 1. Defines state (0 or 1) of LGPL2 during bus clock quarter phases 1 and 2 (first half phase).
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).

Table 12-30. RAM Word Field Descriptions (continued)

Bits	Name	Description
16	G3T1	General-purpose line 3 timing 1. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 1 and 2 (first half phase).
17	G3T3	General-purpose line 3 timing 3. Defines the state (0 or 1) of LGPL3 during bus clock quarter phases 3 and 4 (second half phase).
18	G4T1/DLT3	General-purpose line 4 timing 1/delay time 3. The function of this bit is determined by M*x*MR[GPL4]. If M*x*MR[GPL4] = 0 and LGPL4/LUPWAIT pin functions as an output (LGPL4), G4T1/DLT3 defines the state (0 or 1) of LGPL4 during bus clock quarter phases 1 and 2 (first half phase). If M*x*MR[GPL4] = 1 and LGPL4/LUPWAIT functions as an input (LUPWAIT), if a read burst or single read is executed, G4T1/DLT3 defines the sampling of the data bus as follows: 0 In the current word, the data bus should be sampled at the start of bus clock quarter phase 1 of the next bus clock cycle. 1 In the current word, the data bus should be sampled at the start of bus clock quarter phase 3 of the current bus clock cycle.
19	G4T3/WAEN	General-purpose line 4 timing 3/wait enable. Bit function is determined by M*x*MR[GPL4]. If M*x*MR[GPL4] = 0 and LGPL4/LUPWAIT pin functions as an output (LGPL4), G4T3/WAEN defines the state (0 or 1) of LGPL4 during bus clock quarter phases 3 and 4 (second half phase). If M*x*MR[GPL4] = 1 and LGPL4/LUPWAIT functions as an input (LUPWAIT), G4T3/WAEN is used to enable the wait mechanism: 0 LUPWAIT detection is disabled. 1 LUPWAIT is enabled. If LUPWAIT is detected as being asserted, a freeze in the external signals logical values occurs until LUPWAIT is detected as being negated.
20	G5T1	General-purpose line 5 timing 1. Defines the state (0 or 1) of LGPL5 during bus clock quarter phases 1 and 2 (first half phase).
21	G5T3	General-purpose line 5 timing 3. Defines the state (0 or 1) of LGPL5 during bus clock quarter phases 3 and 4 (second half phase).
22–23	REDO	Redo current RAM word. Defines the number of times to execute the current RAM word. 00 Once (normal operation) 01 Twice 10 Three times 11 Four times
24	LOOP	Loop start/end. The first RAM word in the RAM array where LOOP is 1 is recognized as the loop start word. The next RAM word where LOOP is 1 is the loop end word. RAM words between, and including the start and end words, are defined as part of the loop. The number of times the UPM executes this loop is defined in the corresponding loop fields of the M*x*MR. 0 The current RAM word is not the loop start word or loop end word. 1 The current RAM word is the start or end of a loop.

Table 12-30. RAM Word Field Descriptions (continued)

Bits	Name	Description
25	EXEN	Exception enable. Allows branching to an exception pattern at the exception start address (EXS). When an internal bus monitor time-out exception is recognized and EXEN in the RAM word is set, the UPM branches to the special exception start address (EXS) and begins operating as the pattern defined there specifies. The user should provide an exception pattern to negate signals controlled by the UPM in a controlled fashion. For DRAM control, a handler should negate RAS and CAS to prevent data corruption. If EXEN = 0, exceptions are ignored by UPM (but not by Local Bus) and execution continues. After the UPM branches to the exception start address, it continues reading until the LAST bit is set in the RAM word. 0 The UPM continues executing the remaining RAM words, ignoring any internal bus monitor time-out. 1 The current RAM word allows a branch to the exception pattern after the current cycle if an exception condition is detected.
26–27	AMX	Address multiplexing. Determines the source of LAD[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]. 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*n*[PS], the increment value of the state 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 M*x*MR[DS*n*]. The disable timer does not affect memory accesses to different banks. Note that TODT must be set together with LAST, otherwise it is ignored. 0 The disable timer is turned off. 1 The disable timer for the current bank is activated preventing a new access to the same bank (when controlled by the UPMs) until the disable timer expires. For example, precharge time.
31	LAST	Last word. When LAST is read in a RAM word, the current UPM pattern terminates and control signal timing set in the RAM word is applied to the current (and last) cycle. However, if the disable timer is activated and the next access is to the same bank, execution of the next UPM pattern is held off and the control signal values specified in the last word are extended in duration for the number of clock cycles specified in M*x*MR[DS*n*]. 0 The UPM continues executing RAM words. 1 Indicates the last RAM word in the program. The service to the UPM request is done after this cycle concludes.

12.4.4.4.2 Chip-Select Signal Timing (CST*n*)

If BR*n*[MSEL] of the accessed bank selects a UPM on the currently requested cycle, the UPM manipulates the $\overline{\text{LCS}n}$ for that bank with timing as specified in the UPM RAM word CST*n* fields. The selected UPM affects only the assertion and negation of the appropriate $\overline{\text{LCS}n}$ signal. The state of the selected $\overline{\text{LCS}n}$ signal of the corresponding bank depends on the value of each CST*n* bit. Figure 12-59 shows how UPMs control $\overline{\text{LCS}n}$ signals.

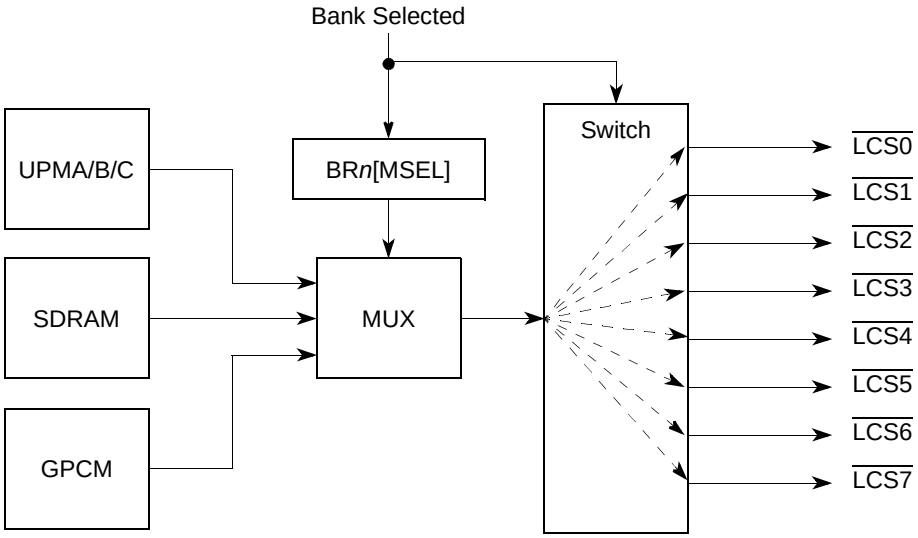


Figure 12-59. $\overline{\text{LCS}n}$ Signal Selection

12.4.4.4.3 Byte Select Signal Timing (BST*n*)

If BR*n*[MSEL] of the accessed memory bank selects a UPM on the currently requested cycle, the selected UPM affects the assertion and negation of the appropriate $\overline{\text{LBS}}$[0:3] signal. The timing of all four byte-select signals is specified in the RAM word. However, $\overline{\text{LBS}}$[0:3] are also controlled by the port size of the accessed bank, the number of bytes to transfer, and the address accessed. Figure 12-60 shows how UPMs control $\overline{\text{LBS}}$[0:3].

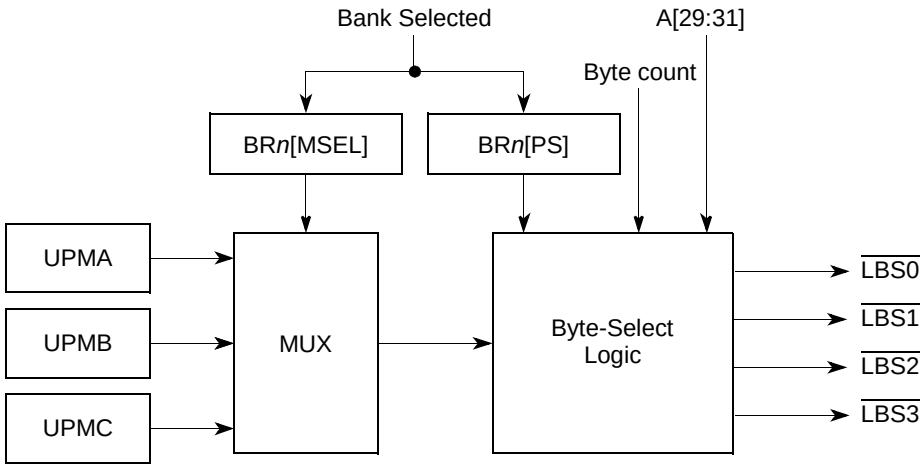


Figure 12-60. $\overline{\text{LBS}}$ Signal Selection

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 (G*n*T*n*, GO*n*)

The general-purpose signals (LGPL[0:5]) each have two bits in the RAM word that define the logical value of the signal to be changed at the rising edge of the bus clock and/or at the falling edge of the bus clock. LGPL0 offers enhancements beyond the other LGPL*n* lines.

GPL0 can be controlled by an address line specified in M*x*MR[G0CL]. To use this feature, G0H and G0L should be set in the RAM word. For example, for a SIMM with multiple banks, this address line can be used to switch between internal memory device banks.

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 if LAST and LOOP must not be set together.

Table 12-31. M*x*MR Loop Field Use

Request Serviced	Loop Field
Read single-beat cycle	RLF
Read burst cycle	RLF
Write single-beat cycle	WLF
Write burst cycle	WLF
Refresh timer expired	TLF
RUN command	RLF

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 more times, as defined in the REDO field of the current RAM word.

Special care must be taken in the following cases:

- When UTA and REDO are set together, TA is asserted the number of times specified by the REDO function.
- When NA and REDO are set together, the address is incremented the number of times specified by the REDO function.
- When LOOP and REDO are set together, the loop mechanism works as usual and the line is repeated according to the REDO function.
- LAST and REDO must not be set together.
- REDO should not be used within the exception routine.

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 M*x*MR[AM] field, or driving the MAR contents on the address signals. In all cases, LA[27:31] of the LBC are driven by the five 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 12-32 shows how M*x*MR[AM] settings affect address multiplexing when the RAM word AMX = 10. The 16 msbs 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

AM	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 OR*n* 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 the LBC), the value of the DLT3 bit in the same RAM word, in conjunction with M*x*MR[GPL*n*4DIS], determines when the data input is sampled by the LBC as follows:

- If M*x*MR[GPL*n*4DIS] = 1 (G4T4/DLT3 functions as DLT3) and DLT3 = 1 in the RAM word, data is latched on the falling edge of the bus clock instead of the rising edge. The LBC samples the data on the next falling edge of the bus clock, which is during the middle of the current bus cycle. This feature should be used only in systems without external synchronous bus devices that require mid-cycle sampling.
- If GPL*n*4DIS = 0 (G4T4/DLT3 functions as G4T4), or if GPL*n*4DIS = 1 but DLT3 = 0, data is latched on the rising edge of the bus clock, which occurs at the end of the current bus clock cycle (normal operation).

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

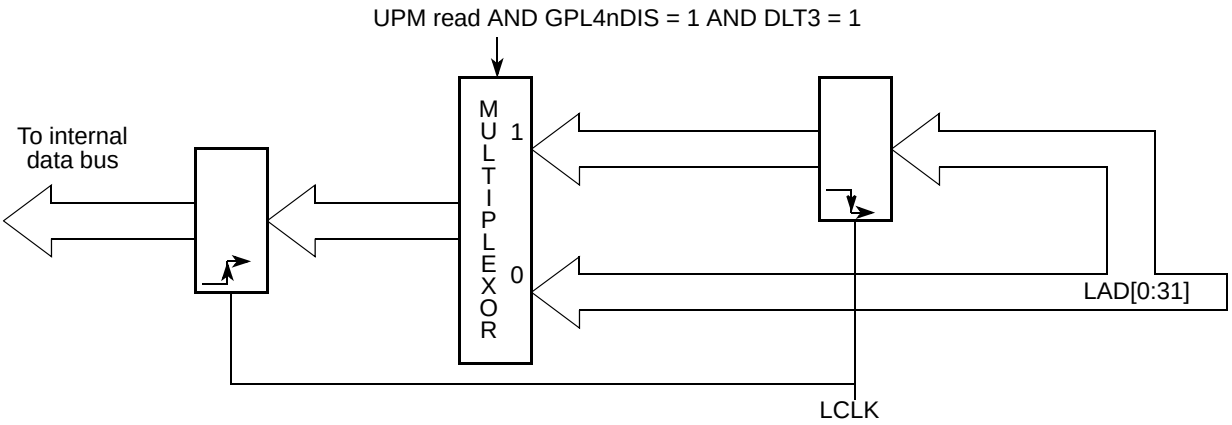


Figure 12-61. UPM Read Access Data Sampling

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 Wait Mechanism (WAEN)

The WAEN bit in the RAM array word can be used to enable the UPM wait mechanism in selected UPM RAM words. If the UPM reads a RAM word with WAEN set, the external LUPWAIT signal is sampled and synchronized by the memory controller as if it were an asynchronous signal. The WAEN bit is ignored if LAST = 1 in the same RAM word.

Synchronization of LUPWAIT starts at the rising edge of the bus clock and takes at least 1 bus cycle to complete. If LUPWAIT is asserted and WAEN = 1 in the current UPM word, the UPM is frozen until LUPWAIT is negated. The value of external signals driven by the UPM remains as indicated in the previous RAM word. When LUPWAIT is negated, the UPM continues normal functions. Note that during WAIT cycles, the UPM does not handle data.

Figure 12-62 shows how the WAEN bit in the word read by the UPM and the LUPWAIT signal are used to hold the UPM in a particular state until LUPWAIT is negated. As the example shows, the $\overline{\text{LCS}n}$ and $\overline{\text{LGPL1}}$ states and the WAEN value are frozen until LUPWAIT is recognized as negated. WAEN is typically set before the line that contains UTA = 1. Note that if WAEN and NA are both set in the same RAM word, NA causes the burst address to increment once as normal regardless of whether the UPM freezes.

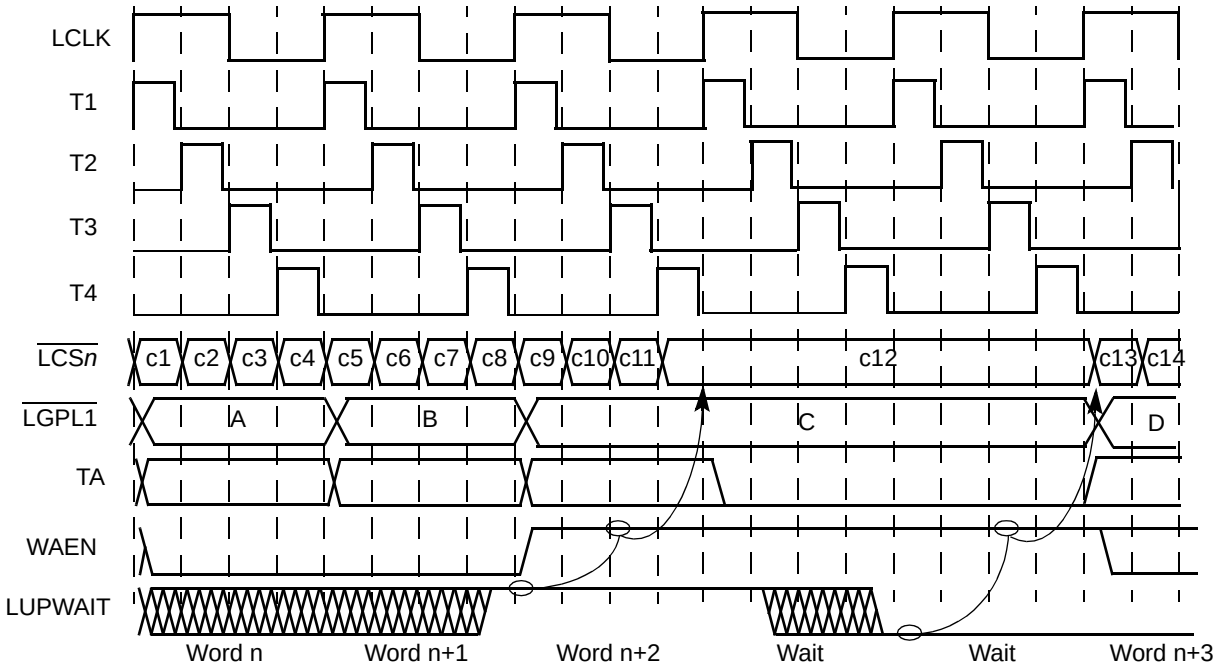


Figure 12-62. Effect of LUPWAIT Signal

12.4.4.5 Synchronous Sampling of LUPWAIT for Early Transfer Acknowledge

If LUPWAIT is to be considered an asynchronous signal, which can be asserted/negated at any time, no UPM RAM word must contain both WAEN = 1 and UTA = 1 simultaneously.

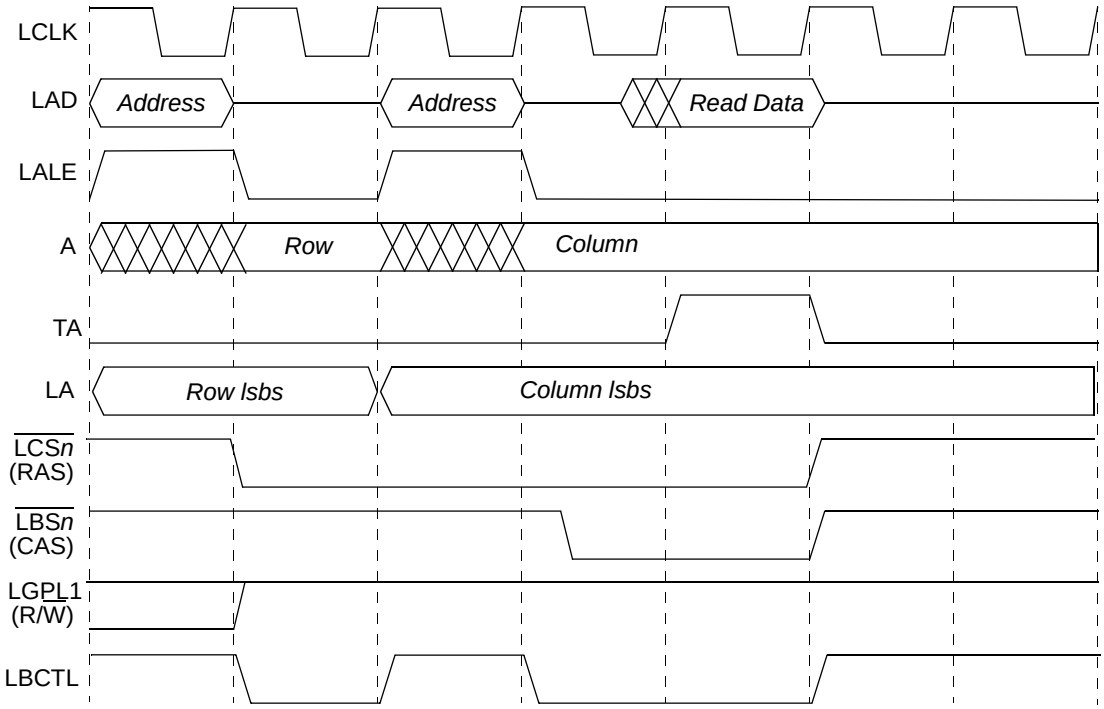
However, programming WAEN = 1 and UTA = 1 in the same RAM word allows UPM to treat LUPWAIT as a synchronous signal, which must meet set-up and hold times in relation to the rising edge of the bus clock. In this case, as soon as UPM samples LUPWAIT negated on the rising edge of the bus clock, it immediately generates an internal transfer acknowledge, which allows a data transfer one bus clock cycle later. The generation of transfer acknowledge is early because LUPWAIT is not re-synchronized, and the acknowledge occurs regardless of whether UPM was already frozen in WAIT cycles or not. This feature allows the synchronous negation of LUPWAIT to affect a data transfer, even if UTA, WAEN, and LAST are set simultaneously.

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 OR*n*[TRLX] and OR*n*[EHTR]. The next accesses after a read access to the slow memory device is delayed by the number of clock cycles specified in the OR*n* register in addition to any existing bus turn around cycle.

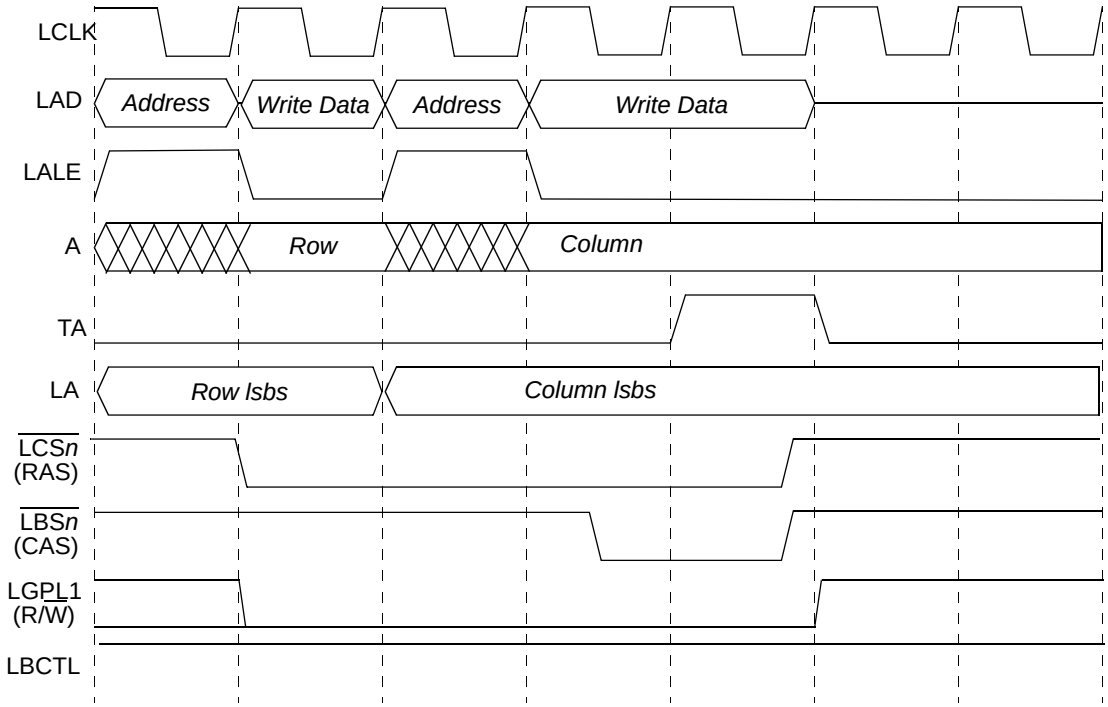
12.4.4.7 Memory System Interface Example Using UPM

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. These illustrative examples may not represent the timing necessary for any specific device used with the LBC. Here, LGPL1 is programmed to drive R/$\overline{W}$ of the DRAM, although any LGPL*n* signal may be used for this purpose.



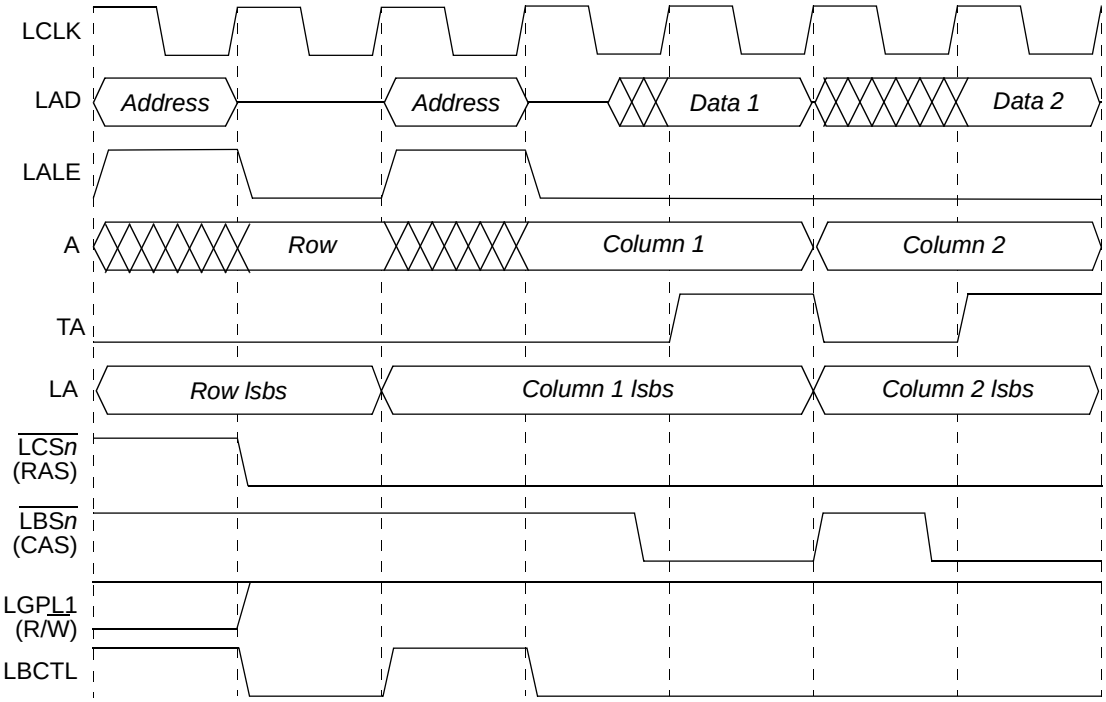
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 0	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 0 0 0	0 0 0 0	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1					Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	1 1		1 1	1 1	Bit 12 Bit 13
g2t1 g2t3					Bit 14 Bit 15
g3t1 g3t3					Bit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

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



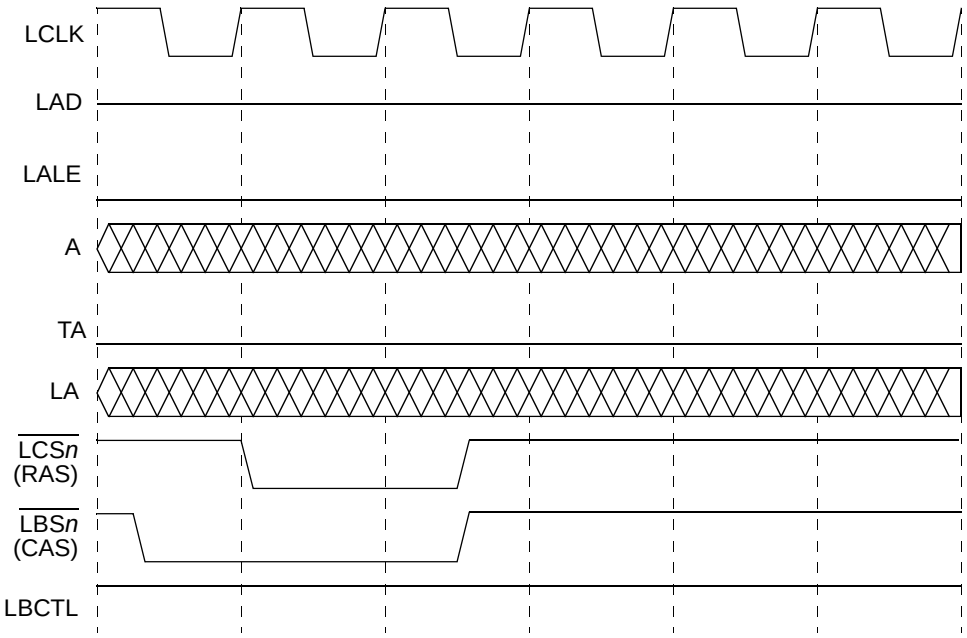
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 1 0 0	0 0 0 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1					Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	0 0		0 0	0 0	Bit 12 Bit 13
g2t1 g2t3					Bit 14 Bit 15
g3t1 g3t3					Bit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

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



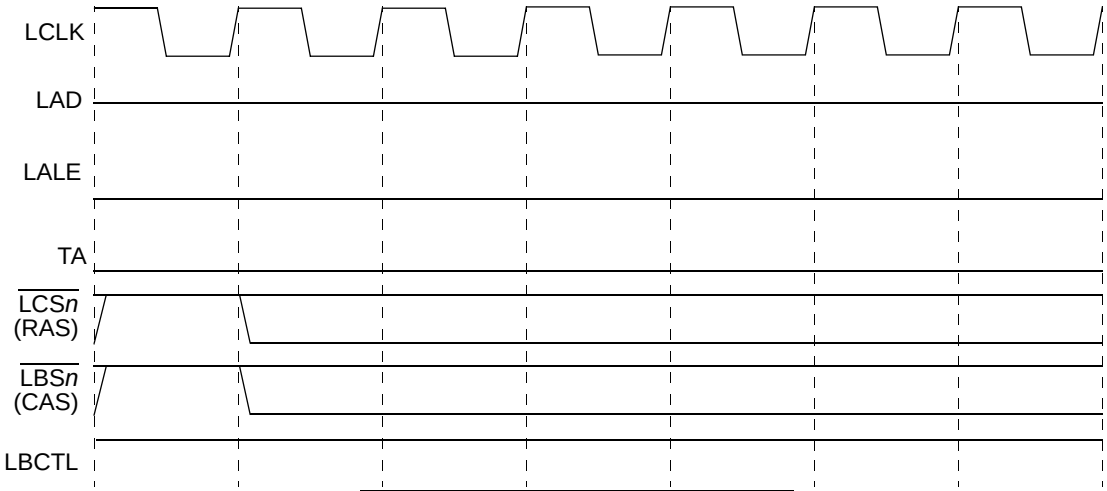
cst1 cst2 cst3 cst4	0 0 0 0	LALE pause (due to change in AMX)	0 0 0 0	0 0 0 0	1 1 1 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1		1 1 1 0	0 0 0 0	1 1 1 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1						Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3	1 1		1 1	1 1	1 1	Bit 12 Bit 13
g2t1 g2t3						Bit 14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

Figure 12-65. Burst Read Access to FPM DRAM Using LOOP (Two 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 cst2 cst3 cst4	1 1 1 1	Bit 0 Bit 1 Bit 2 Bit 3
bst1 bst2 bst3 bst4	1 1 1 1	Bit 4 Bit 5 Bit 6 Bit 7
g0l0 g0l1 g0h0 g0h1		Bit 8 Bit 9 Bit 10 Bit 11
g1t1 g1t3		Bit 12 Bit 13
g2t1 g2t3		Bit 14 Bit 15
g3t1 g3t3		Bit 16 Bit 17
g4t1 g4t3		Bit 18 Bit 19
g5t1 g5t3		Bit 20 Bit 21
redo[0] redo[1]		Bit 22 Bit 23
loop exen	0 0	Bit 24 Bit 25
amx0 amx1 na uta todt last	0 0 0 0 1 1	Bit 26 Bit 27 Bit 28 Bit 29 Bit 30 Bit 31
	EXS	

Figure 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 demultiplex and reconstruct the original address. No external intelligence is needed, because the LALE signal provides the correct timing to control a standard logic latch. The LAD pins can be directly connected to the data signals of the memory/peripheral.

Transactions on the local bus start with an address phase, where the LBC drives the transaction address on the LAD signals and asserts the LALE signal. This can be used to latch the address and then the LBC can continue with the data phase.

The LBC supports port sizes of 8,16, and 32 bit. For devices smaller than 32 bits, transactions must be broken down. For this reason, LA[30:31] are driven unmultiplexed. For 8-bit devices, LA[30:31] should be used and for 16-bit devices, LA[30] should be used. 32-bit devices use neither of these signals.

In addition, the LBC supports burst transfers (not in the GPCM machine). LA[27:29] are the burst addresses within a natural 32-byte burst. To minimize the amount of address phases needed on the local bus and to optimize the throughput, those signals are driven separately and should be used whenever a device requires the 5 least significant addresses. Those should not be used from LAD[27:31].

All other addresses, A[0:26], must be reconstructed through the latch.

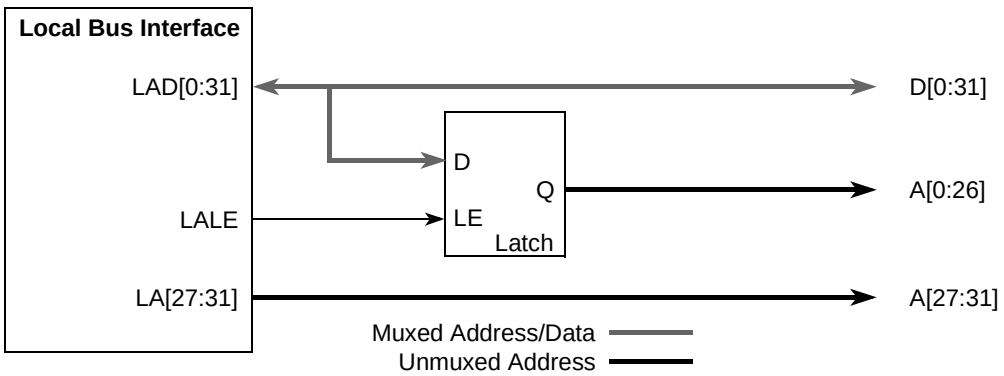


Figure 12-68. Multiplexed Address/Data Bus

12.5.1.2 Peripheral Hierarchy on the Local Bus

To achieve high bus speed interfaces for synchronous SRAMs or SDRAMs, a hierarchy of the memories/peripherals connected to the local bus is suggested, as shown in Figure 12-69.

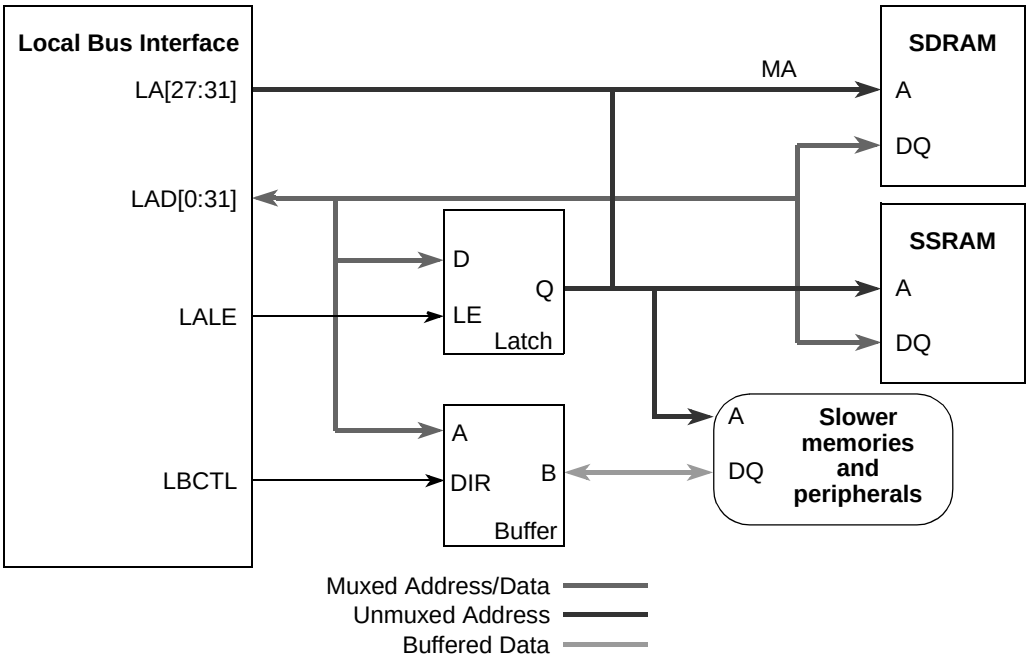


Figure 12-69. Local Bus Peripheral Hierarchy

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 further. For those cases probably only one bank of synchronous SRAMs or SDRAMs should be used and instead of using a separate latch and a separate bus transceiver, a bus demultiplexer combining those two functions into one device should be used. Figure 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.

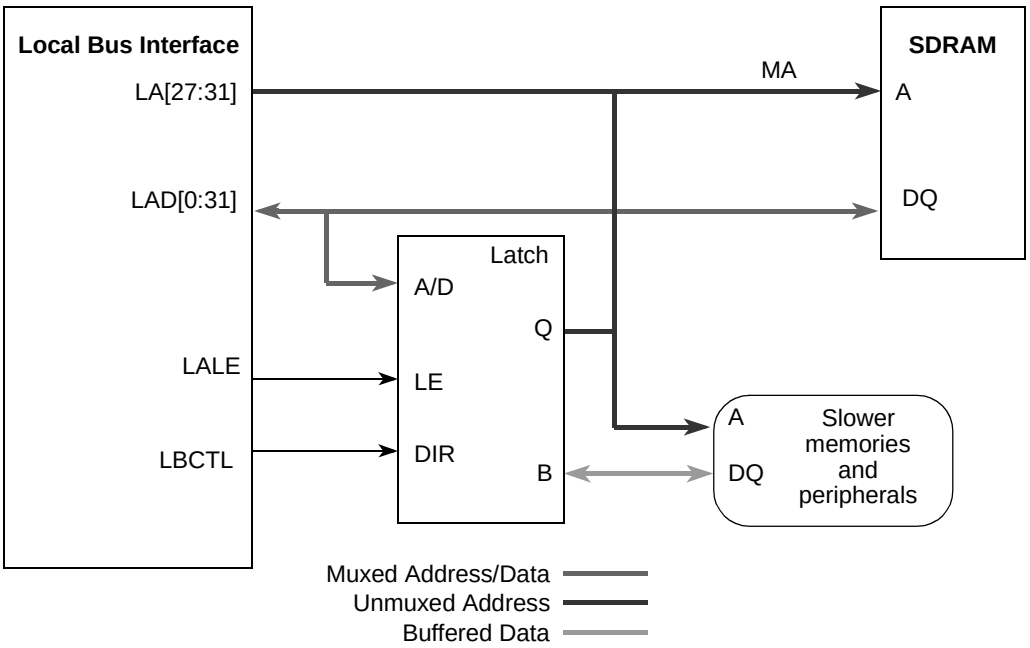


Figure 12-70. Local Bus Peripheral Hierarchy for Very High Bus Speeds

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 (for example, FLASH EPROM and peripherals) the GPCM-controlled memories should be decoupled by buffers to reduce capacitive loading on the bus. Those buffers have to be taken into account for the timing calculations.

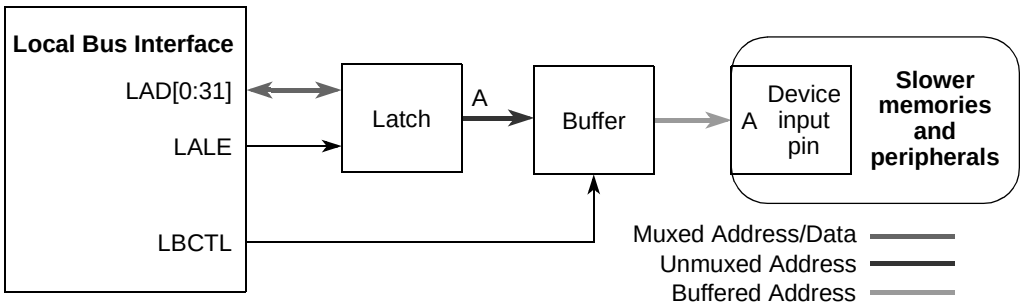


Figure 12-71. GPCM Address Timings

To calculate address setup timing for a slower peripheral/memory device, several parameters have to be added: propagation delay for the address latch, propagation delay for the buffer and the address setup for the actual peripheral. Typical values for the two propagation delays are in the order of 3–6 ns, so for a 166 MHz bus frequency, $\overline{\text{LCS}}$ should arrive on the order of 3 bus clocks later.

For data timings, only the propagation delay of one buffer plus the actual data setup time has to be considered.

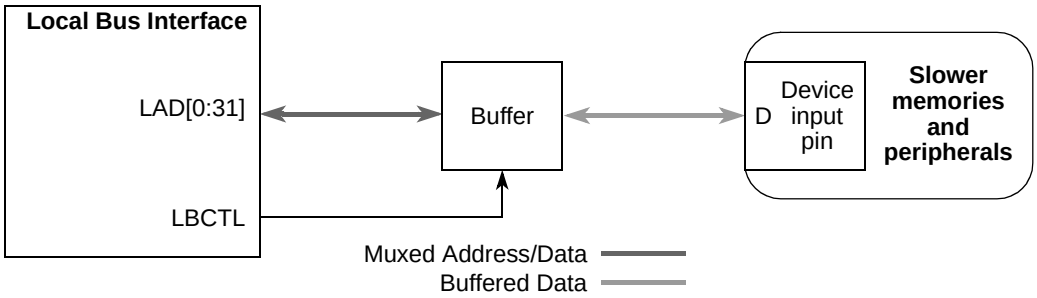


Figure 12-72. GPCM Data timings

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

The bus does not change direction for the following cases so they need no special attention:

- Continued burst after the first beat
- Write data phase after address phase
- Address phase after previous write

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 impedance 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 the bus transceivers in the same direction as during a write. After the end of the address phase, LBCTL goes low and changes the direction of the bus transceiver. The 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.

However, bus contention could potentially still occur on the far side of a bus transceiver. It is the responsibility of the designer of the UPM pattern to guarantee that enough idle cycles are inserted in the UPM pattern to avoid this.

12.5.3 Interface to Different Port-Size Devices

The LBC supports 8-, 16-, and 32-bit data port sizes. However, the bus requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 32-bit port must reside on 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.

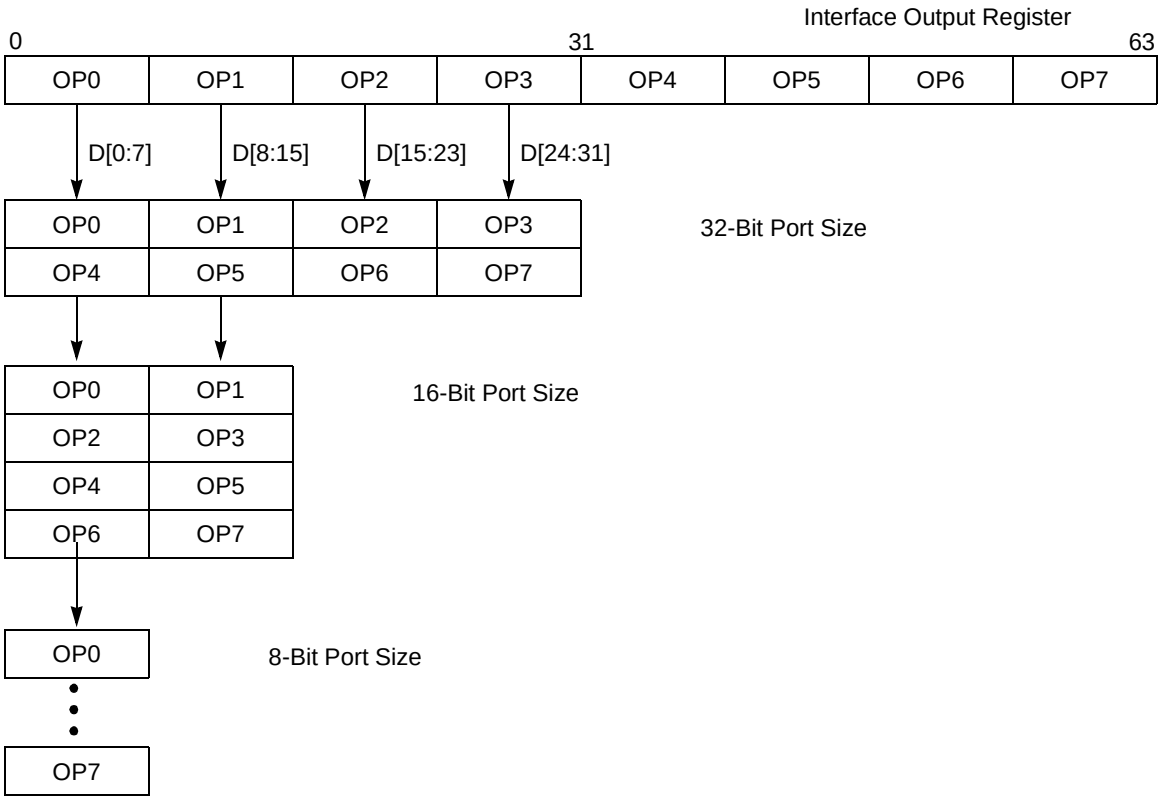


Figure 12-73. Interface to Different Port-Size Devices

Table 12-33 lists the bytes required on the data bus for read cycles.

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

The following subsections provide application information on 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 ($\overline{\text{LCS}}n$) 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 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 principle 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:

$$\text{Refresh Period} = \frac{\text{LSRTx (MPTPR[PTP])}}{\text{System Frequency}}$$

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 in Section 12.4.3.11.1, "SDRAM 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 Multiplexing

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.

12.5.4.3.3 128-Mbyte SDRAM

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

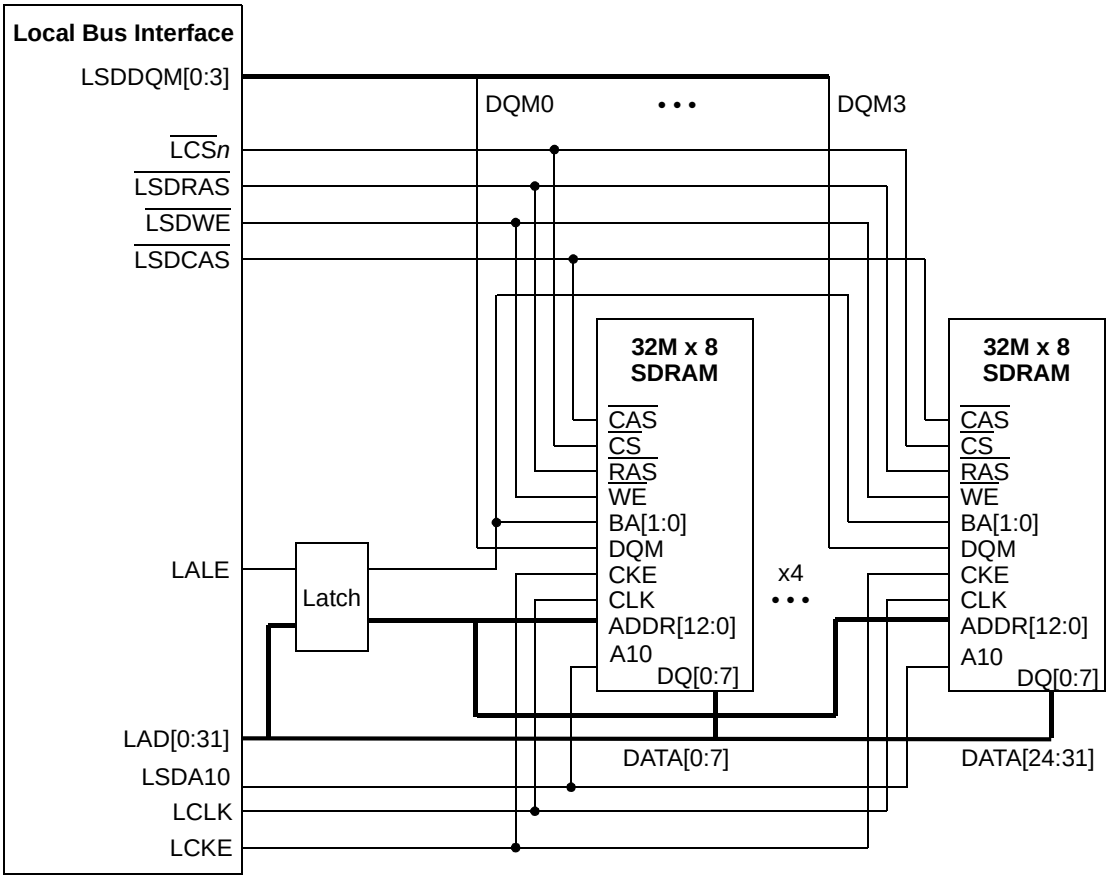


Figure 12-74. 128-Mbyte SDRAM Diagram

Table 12-35 shows details about LAD*n* signal connections for the example in Figure 12-74.

Table 12-35. LAD*n* Signal Connections to 128-Mbyte SDRAM

LAD (latch address)	SDRAM Address Pin
LAD29	A0
LAD28	A1
LAD27	A2
LAD26	A3
LAD25	A4
LAD24	A5
LAD23	A6
LAD22	A7
LAD21	A8
LAD20	A9
LAD19 (no connect)	A10 is connected to LSDA10

Table 12-35. LAD*n* Signal Connections to 128-Mbyte SDRAM (continued)

LAD (latch address)	SDRAM Address Pin
LAD18	A11
LAD17	A12
LAD16	BA0 (if LSDMR[BSMA] = 011)
LAD15	BA1 (if LSDMR[BSMA] = 011)

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 12-36.

Table 12-36. 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 12-37.

Table 12-37. 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 12-38 shows the address port configuration during a READ/WRITE command.

Table 12-38. 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 12-39 shows the register configuration for this example. PSRT and MPTPR are not shown but should be programmed according to the device's specific refresh requirements.

Table 12-39. Register Settings for 128-Mbytes SDRAMs

Register	Field	Value
BR*n*	BA	Base address
	XBA	Ext. Base Address
	PS	11 = 32-bit port size
	MS	011 = SDRAM-local bus
	V	1
OR*n*	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

This example uses the MT48LC64M4A2FB from Micron to implement 512 Mbytes.

In this SDRAM organization:

- The 32-bit port size is $8 \times 4 \times 64$ Mbit $\times$ 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 12-40.

Table 12-40. 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 12-41.

Table 12-41. 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 12-42 shows the address port settings during a READ/WRITE command.

Table 12-42. 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 12-43 shows the register configuration. PSRT and MPTPR are not shown, but they should be programmed according to the specific device's refresh requirements.

Table 12-43. Register Settings for 512-Mbyte SDRAMs

Register	Field	Value
BR*n*	BA	Base address
	XBA	ext. Base address
	PS	11 = 32-bit port size
	MS	011 = SDRAM-local bus
	V	1
OR*n*	AM	11_1110_0000_0000_0000_0
	XAM	11
	BPD	01
	COLS	100
	ROWS	100

Table 12-43. Register Settings for 512-Mbyte SDRAMs (continued)

Register	Field	Value
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, and 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. Note that the figure does not show the precharge-all command that is issued by the LBC automatically prior to the self-refresh command.

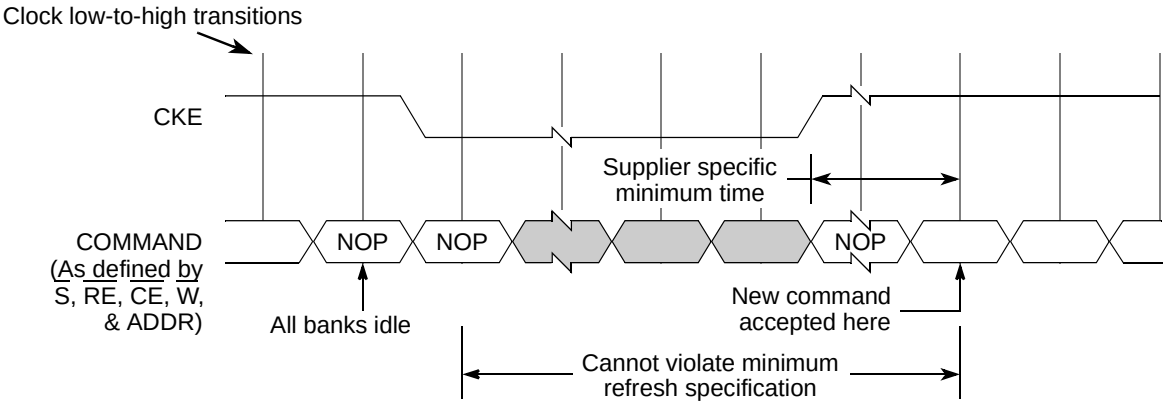


Figure 12-75. SDRAM Power-Down Timing

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 the SDRAM CKE (LCKE) signal 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. As in the case of the power-down mode, the figure does not show the precharge-all command that is issued by the LBC automatically prior to the self-refresh command.

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.

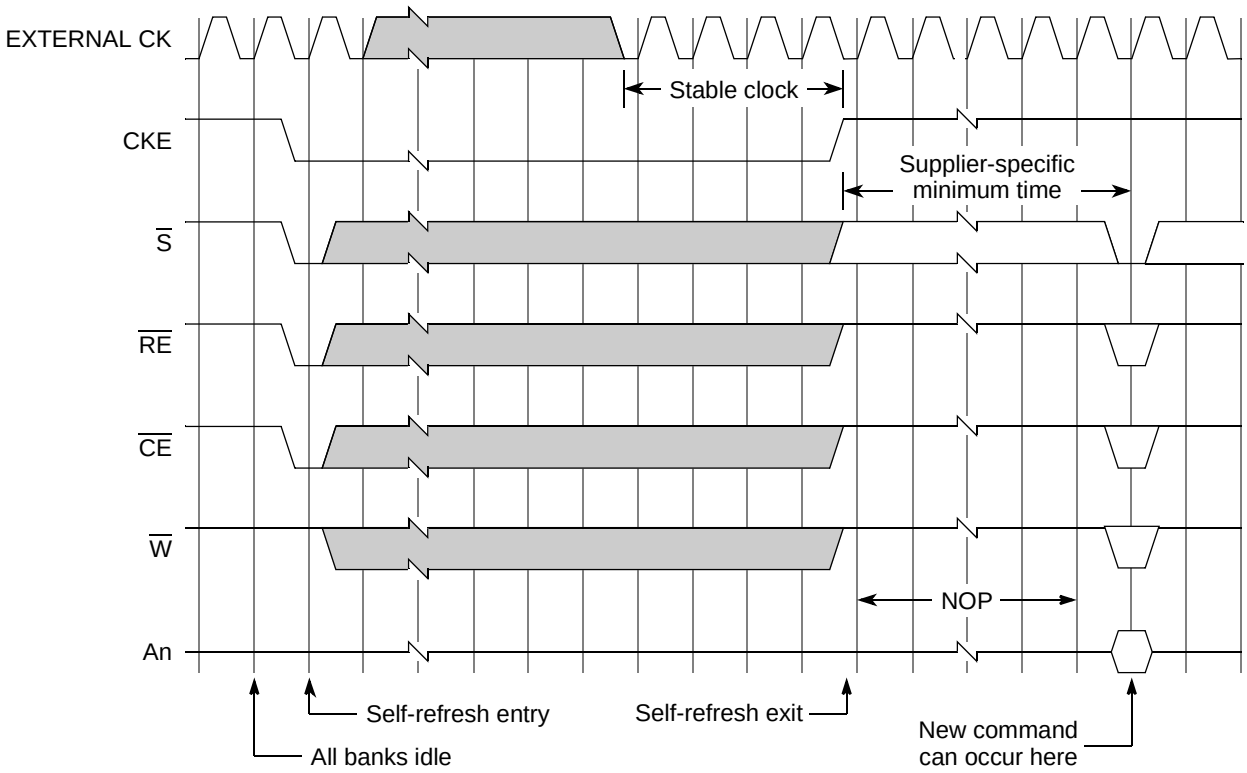


Figure 12-76. SDRAM Self-Refresh Mode Timing

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 as shown in Table 12-44.

Table 12-44. 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 data sheets 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 12-45. 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 data sheets 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, as described in the rest of this subsection.

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 the local bus clock DLL.

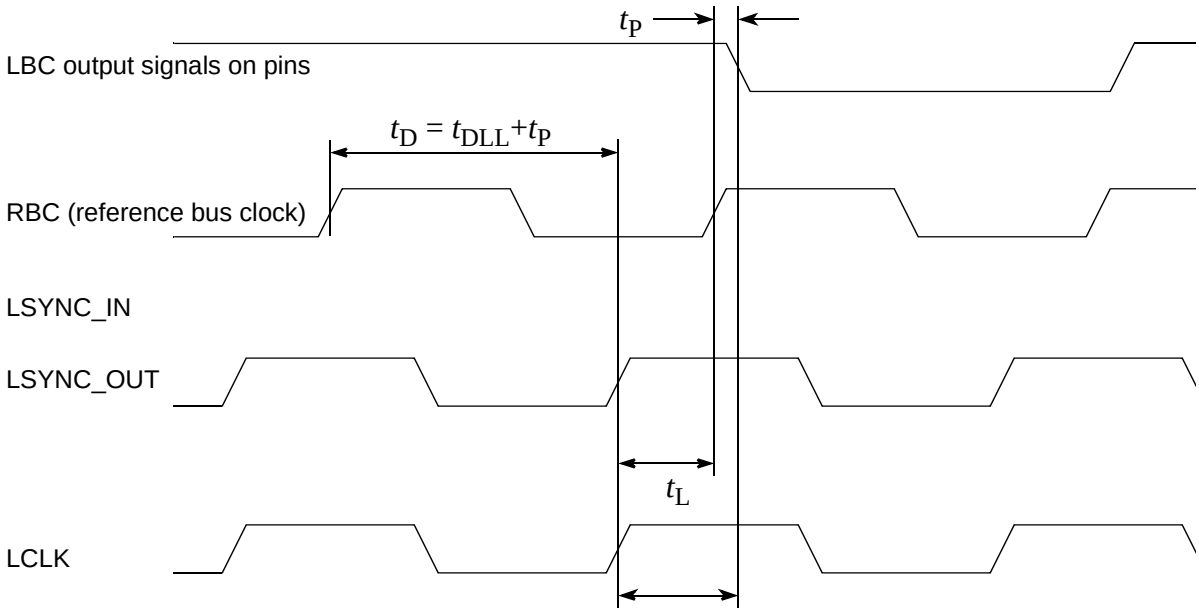


Figure 12-77. Local Bus DLL Operation

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.

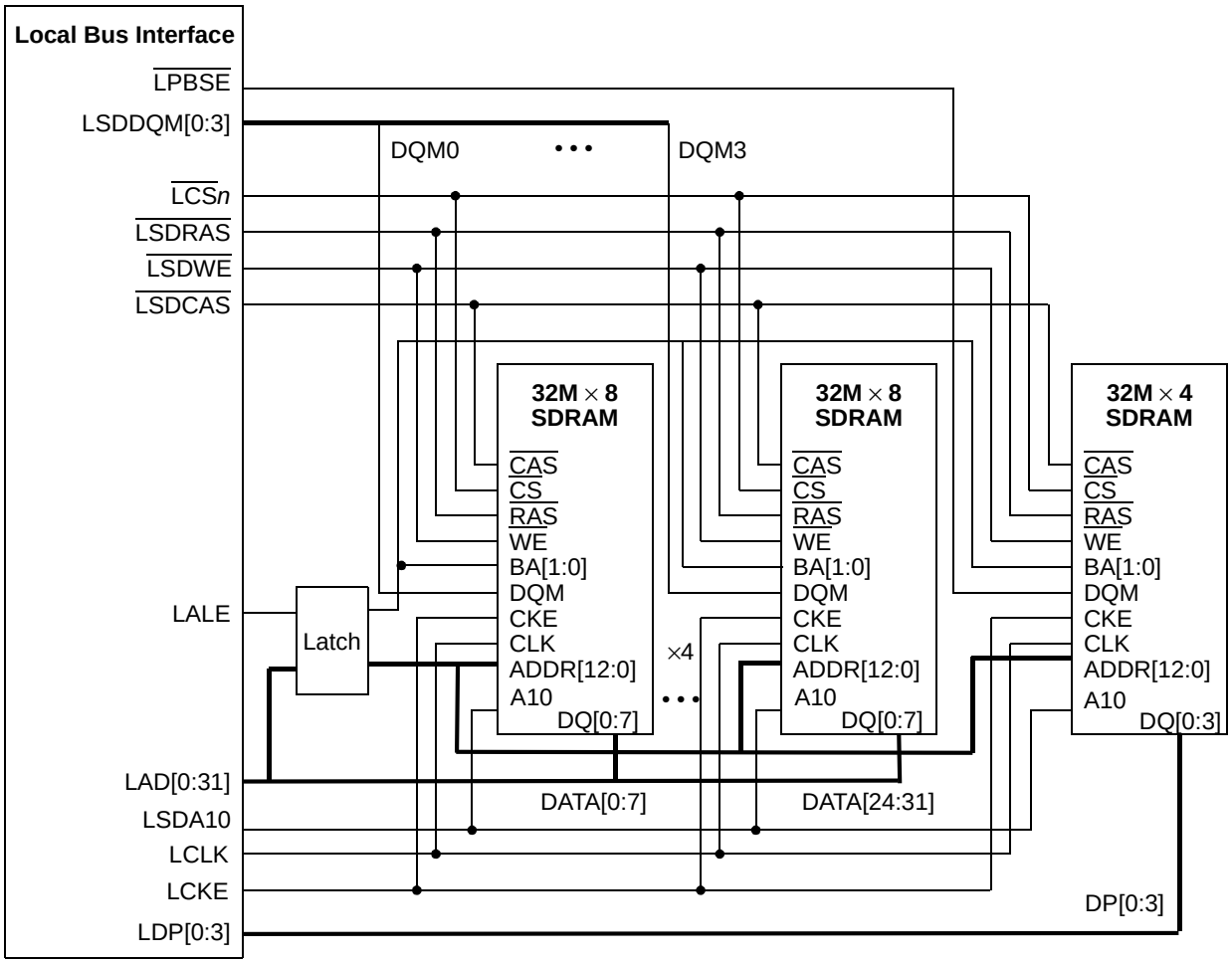


Figure 12-78. Parity Support for SDRAM

12.5.5 Interfacing to ZBT SRAM

In many applications, SDRAM provides sufficient performance for the local bus. However, especially in networking applications, memory access patterns are often random and SDRAM is not 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 is used to generate control signals. The same interfacing is used for pipelined and flow-through versions of ZBT SRAMs. However different UPM patterns must be generated for those cases. Because ZBT SRAMs will mostly be used by performance-critical applications, we assume here that, typically, the maximum width of the local bus of 32 bits will be used.

ZBT SRAMs allow different configurations. For the local bus the burst order should be set to linear burst order by tying the mode pin to GND; $\overline{\text{CKE}}$ should also be tied to ground.

ZBT SRAMs perform four-beat bursts. Because the LBC generates eight-beat transactions (for 32 bit ports) the UPM breaks down each burst into two consecutive four-beat bursts. The internal address generator of the LBC generates the new A27 for the second burst.

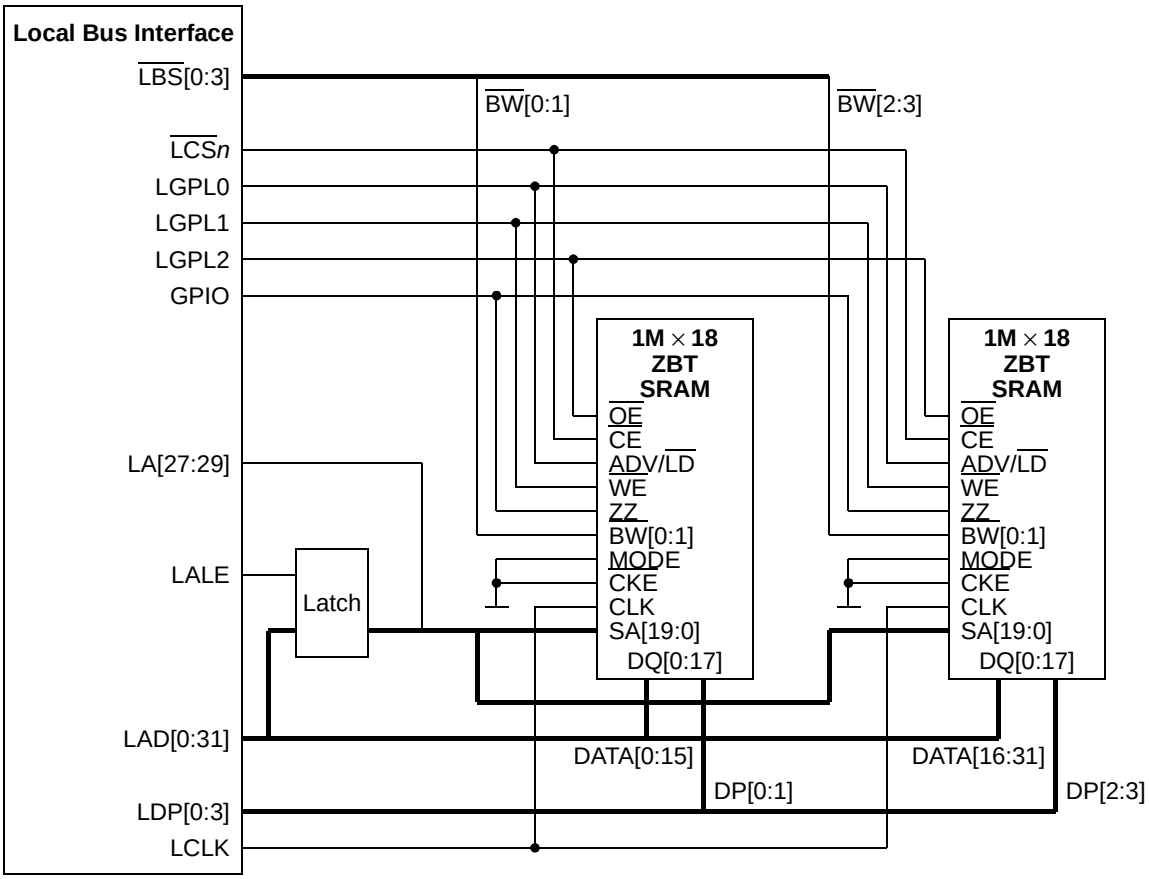


Figure 12-79. Interface to ZBT SRAM

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 ×18 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 interface, the host and HDI16 device can communicate 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. It 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 16-bit Digital Signal Processor Reference Manual* (order number MSC8101RM); 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 four 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 eight 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.

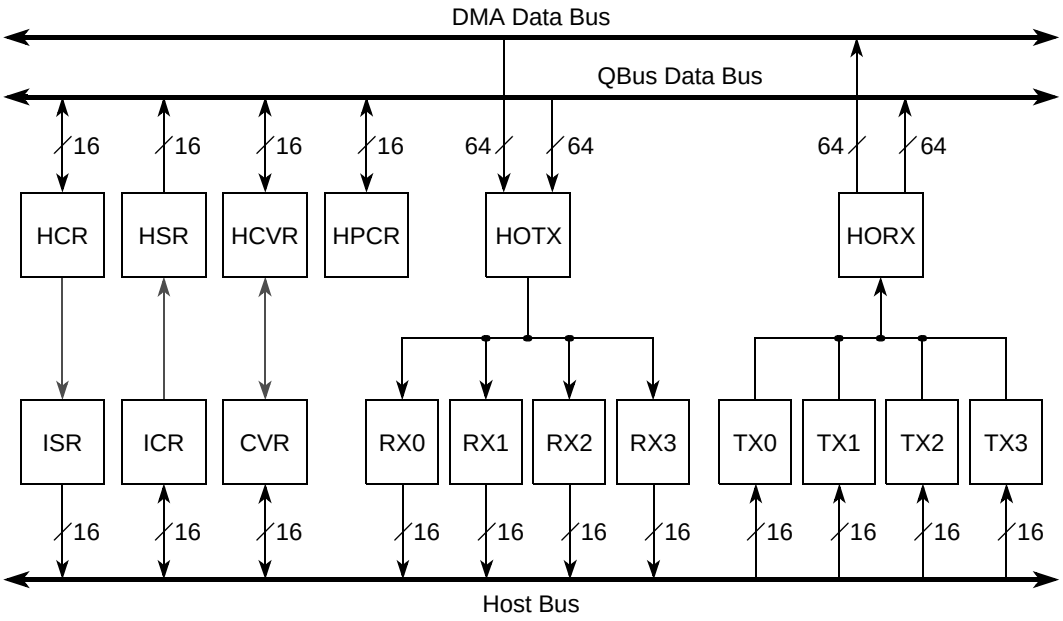


Figure 12-80. MSC8101 HDI16 Peripheral Registers

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 12-46 and Figure 12-81.

Table 12-46. 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]
$\overline{\text{HCS1}}$	I	Host chip select 1	$\overline{\text{LCS}n}$
$\overline{\text{HCS2}}$	I	Host chip select 2	Tie this to V_{DD}
HRDRW	I	Host R/$\overline{\text{W}}$ signal	LGPL1 or inverted LBCTL signal
$\overline{\text{HDS}}$	I	Host data strobe signal	LGPLy
HRRQ/HACK	O	Receive host request OP	As required in application
HTRQ/HREQ	O	Transmit host request OP	As required in application

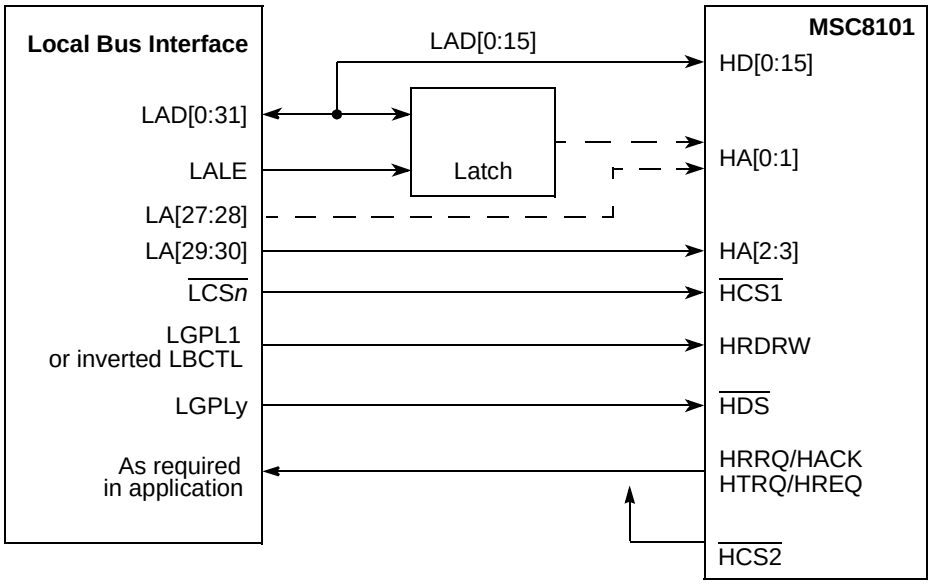


Figure 12-81. Interface to MSC8101 HDI16

The connections are specified as follows:

- One chip select, LCS*n* (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 pattern 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 (LAD*n*) are directly connected to the HDI16 data lines (HD*n*).
- 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 that 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.

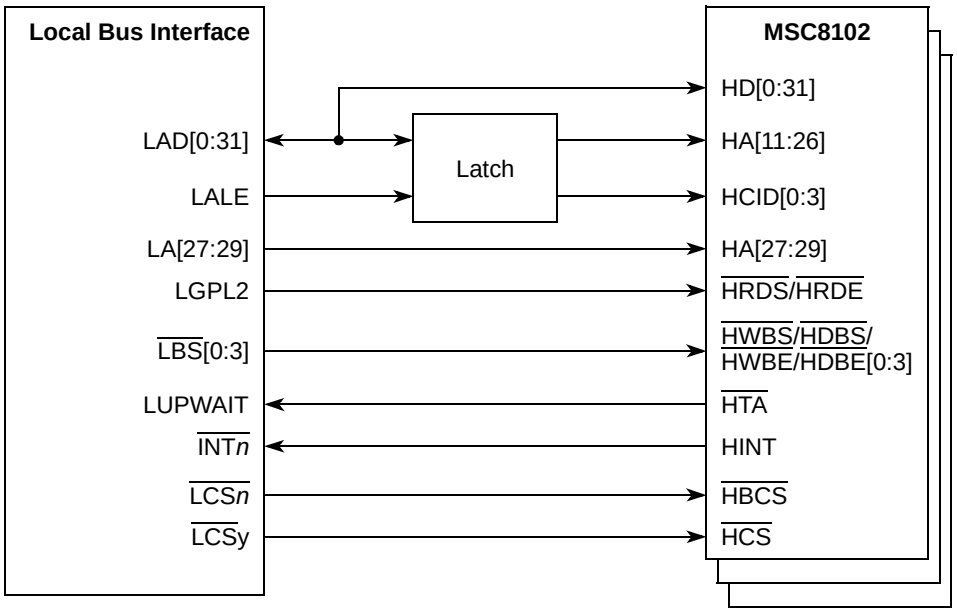


Figure 12-82. Interface to MSC8102 DSI in Asynchronous Mode

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 LBC should be used.

The UPM programmer is responsible for ensuring correct setup and hold timings for all signals. 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 M*x*MR must be cleared to interpret the correct polarity of $\overline{\text{HTA}}$. 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.

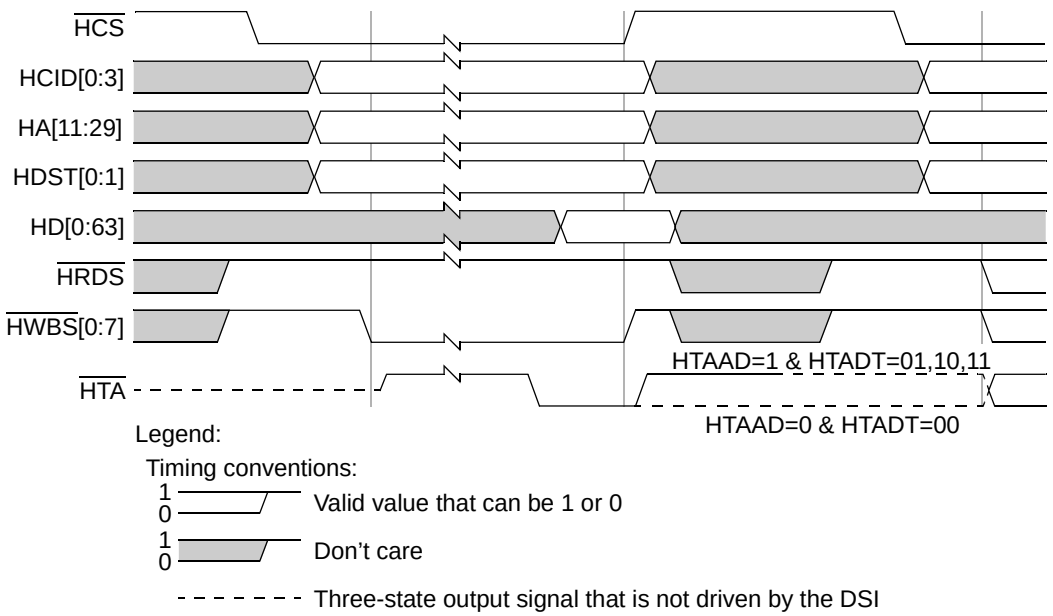


Figure 12-83. Asynchronous Write to MSC8102 DSI

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 HRDS. If the data for this access is already in the read buffer due to the prefetch mechanism, assertion time of HTA is improved. The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. M*x*MR[UWPL] has to be cleared to interpret the correct polarity of the 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 HTA.

The host can start its next access (back-to-back accesses) without negating the HCS signal between accesses. If the next access is not to the same MSC8102, then to prevent contention on HTA, the host must wait until the previous DSI stops driving HTA before it accesses the next device. If the next access is to the same MSC8102, the host must not start consecutive access before HTA 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 is inactive.

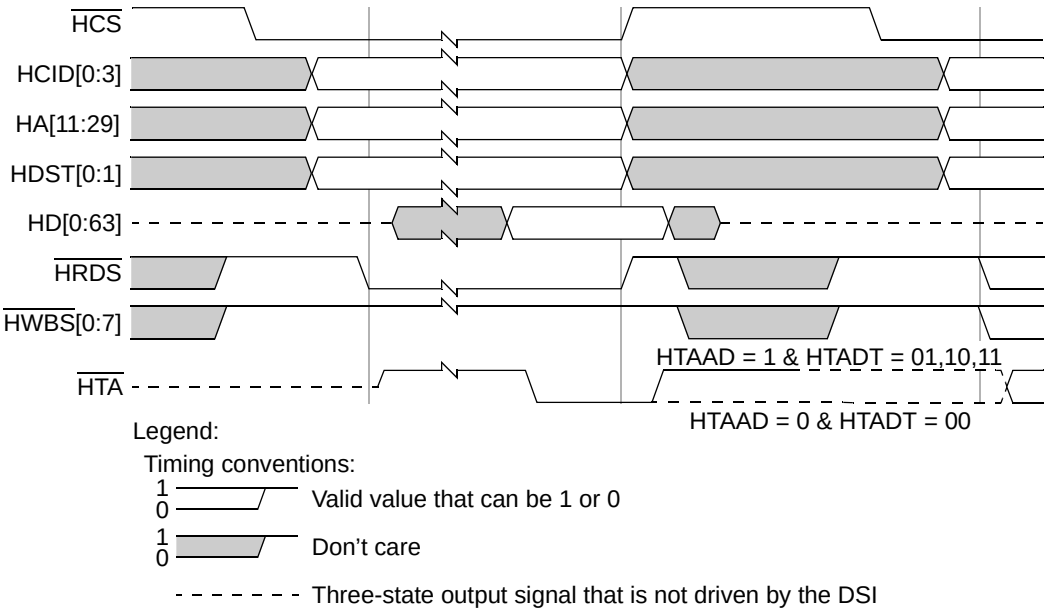


Figure 12-84. Asynchronous Read from MSC8102 DSI

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 LBC must be used to implement this interface.

Figure 12-85 shows the interface for synchronous mode. Because the DSI will assert and negate 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 LUPWAIT of the UPM. M*x*MR[UWPL] must be cleared to interpret the correct polarity of $\overline{\text{HTA}}$. Because this signal influences the internal state machine of the local bus clock, the local bus cannot react to $\overline{\text{HTA}}$ changes correctly within one local bus clock. Refer to Section 12.4.4.4.10, "Wait Mechanism (WAEN)," for more detailed information.

The solution to this lies in that the local bus operates at a higher frequency than the DSI interface of the DSP. The local bus clock can be divided by an integer divider (1:2, 1:3 or 1:4) to generate the DSI clock. This should not be a problem because the local bus is designed for much higher frequencies than the DSI. Because all timings are given in DSP DSI clock cycles, the UPM patterns must be adjusted appropriately and need to assert a signal for 2, 3 or 4 clocks (as many as the divider ratio) instead of one. Fortunately, the UPM has the REDO feature, which allows every UPM RAM entry to be executed 1×, 2×, 3× or 4×, which should be sufficient for any divider ratio that would be used in this case.

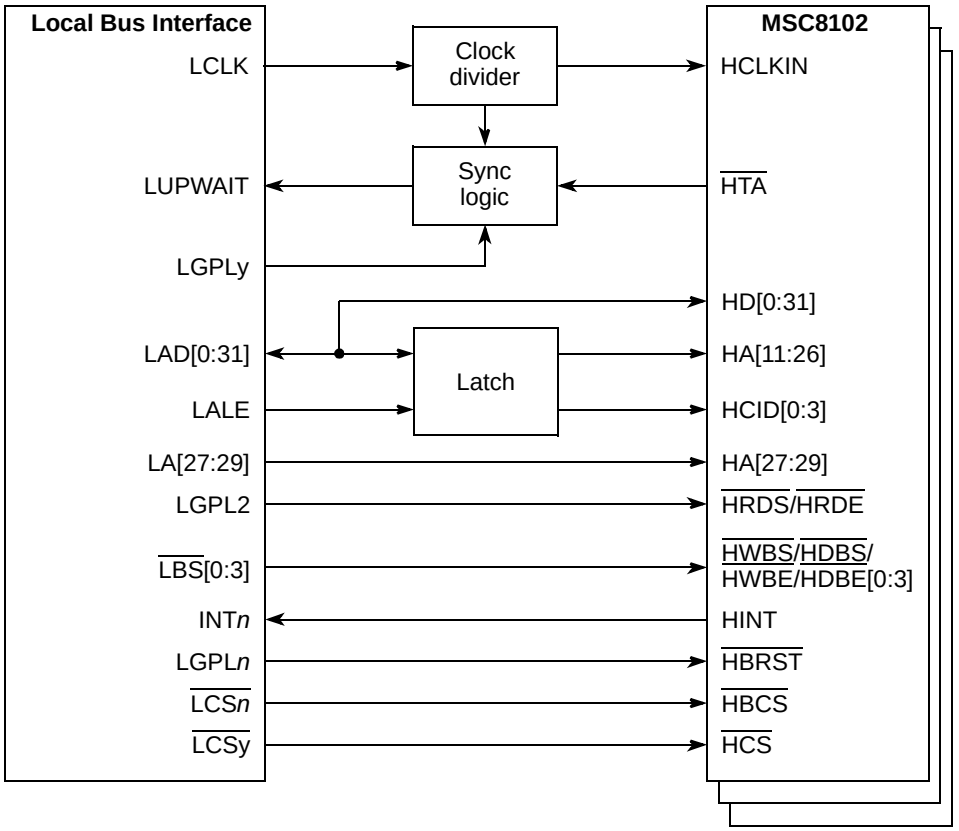


Figure 12-85. Interface to MSC8102 DSI in Synchronous Mode

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, LUPWAIT is synchronized internally, and only 2 clocks after LUPWAIT changes, new data can be sampled or presented. For example, if the local bus

clock ratio is 3× 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 2× the DSI clock, there is a special mode, where the LUPWAIT is NOT synchronized. Refer to Section 12.4.4.5, "Synchronous Sampling of LUPWAIT 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 LUPWAIT must be met in this mode; otherwise indeterministic 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 multiplexer and to activate external synchronization logic, which uses the DSI clock to stall the UPM by asserting LUPWAIT until the beginning of the next DSI cycle. After that, this GPL signal must be negated and the multiplexer connects LUPWAIT 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 LUPWAIT to get in lockstep with the DSI clock. For example, if the clock divider ratio is 1:3 and the LUPWAIT reaction time is two local bus clocks, because LUPWAIT is synchronized, then one local bus clock should be inserted.

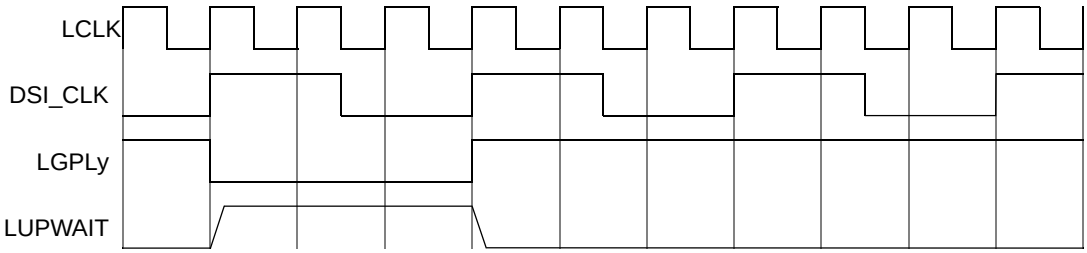


Figure 12-86. UPM Synchronization Cycle

Table 12-47. 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

Table 12-47. UPM Synchronization Cycles (continued)

	Sync CYCLE	Compensation CYCLE	DSI CYCLE 1	Bits
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.

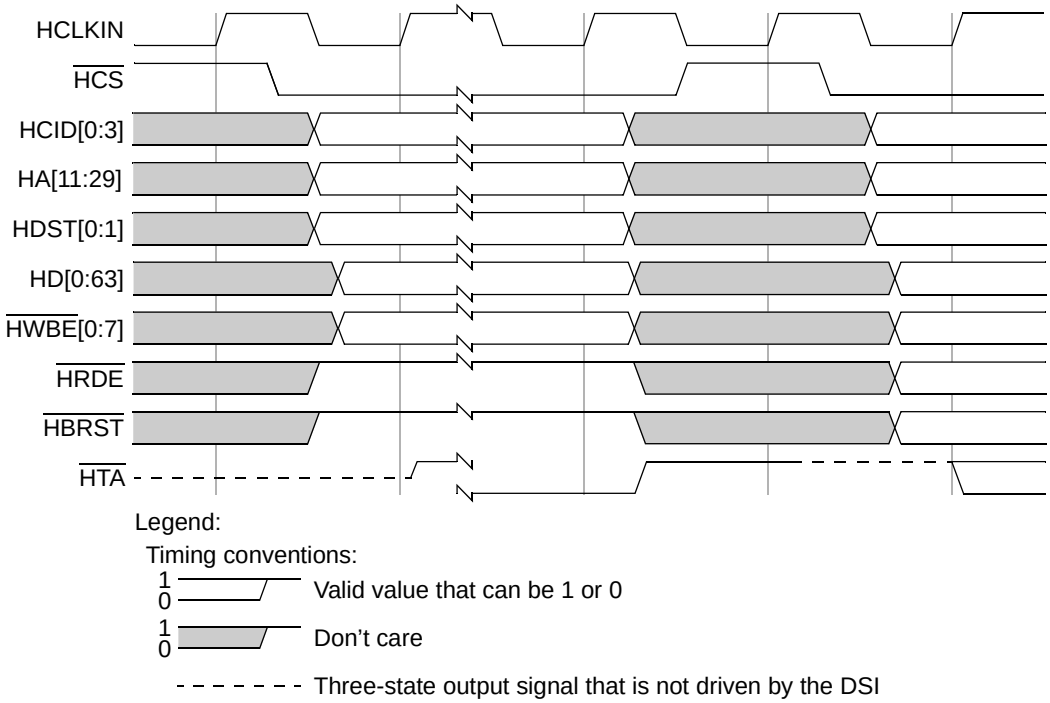


Figure 12-87. Synchronous Single Write to MSC8102 DSI

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. At least one HWBE signal is asserted, and HRDE and HBRST are negated. Assertion of HTA indicates that the DSI is ready to complete the current access and the host must terminate this access. Because HTA is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until HTA goes to 0 and then the UPM continues in its pattern. Typically, HTA is asserted immediately. If the write buffer is full, HTA assertion is delayed. 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 is inactive.

Synchronous Single Read

Figure 12-88 shows a synchronous single read access.

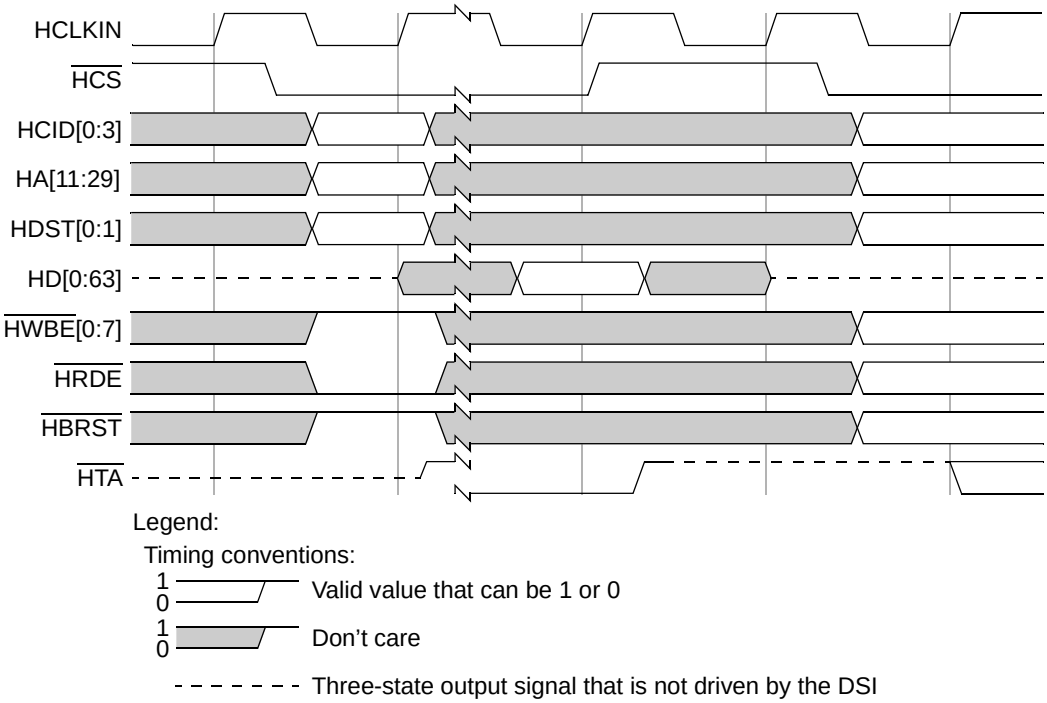


Figure 12-88. Synchronous Single Read from MSC8102 DSI

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. If DCR[8]:RPE 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 UPM LUPWAIT signal, all local bus signals are frozen until HTA goes to 0; then the UPM continues in its pattern. HTA is asserted earlier when the data for this access is already prefetched to the read buffer. It 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 is inactive.

Synchronous Burst Write

Figure 12-89 shows a synchronous burst write access.

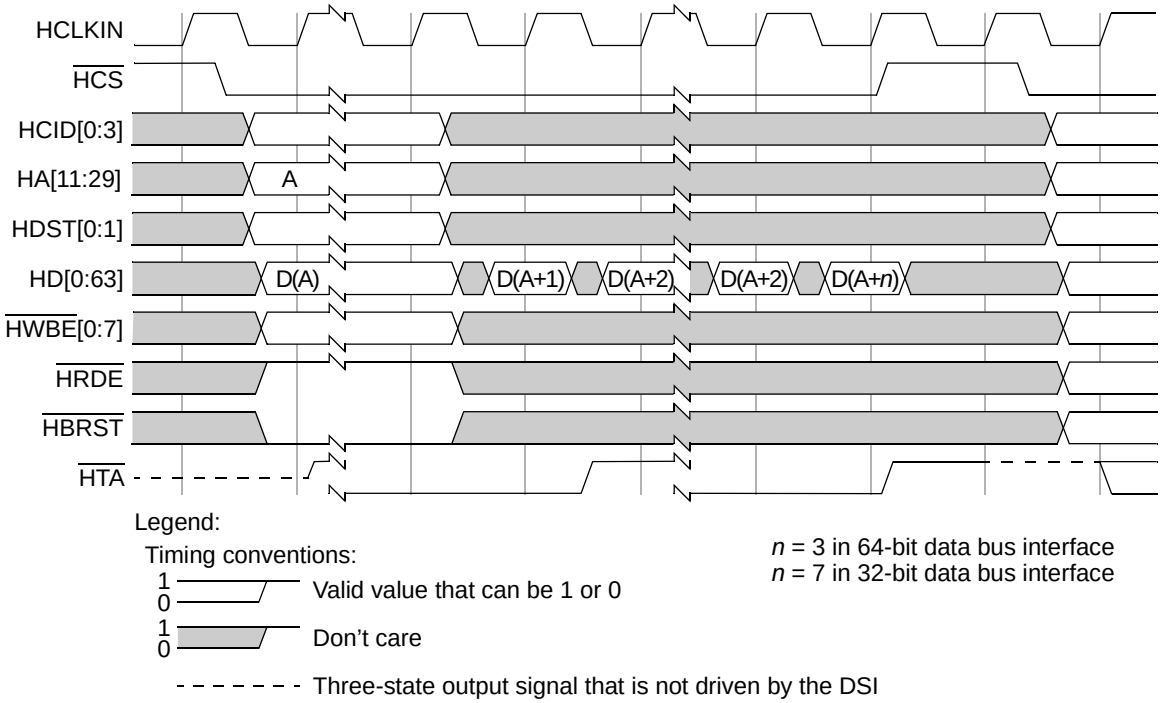


Figure 12-89. Synchronous Burst Write to MSC8102 DSI

The DSI samples HA, HDST, HCID, HD, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. $\overline{\text{HWBE}}$ are asserted, $\overline{\text{HBRST}}$ is asserted, and $\overline{\text{HRDE}}$ is negated. Assertion of $\overline{\text{HTA}}$ indicates that the DSI is ready to complete the current beat of the access and the host must proceed to the next beat of this access. When the host reaches the last beat of the access, it must terminate the burst access. Typically $\overline{\text{HTA}}$ is asserted immediately for each beat of the access. If the write buffer is full, $\overline{\text{HTA}}$ assertion is delayed. Because $\overline{\text{HTA}}$ is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until $\overline{\text{HTA}}$ goes to 0 and then the UPM continues in its pattern. After the last beat of the access, $\overline{\text{HTA}}$ is driven to logic 1 and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8102 immediately in the next HCLKIN rising edge without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8102, then, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving $\overline{\text{HTA}}$. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

Synchronous Burst Read

Figure 12-90 shows a synchronous burst read access. The DSI samples HA, HDST, HCID, $\overline{\text{HWBE}}$, $\overline{\text{HRDE}}$, and $\overline{\text{HBRST}}$ on the first HCLKIN rising edge on which $\overline{\text{HCS}}$ is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed.

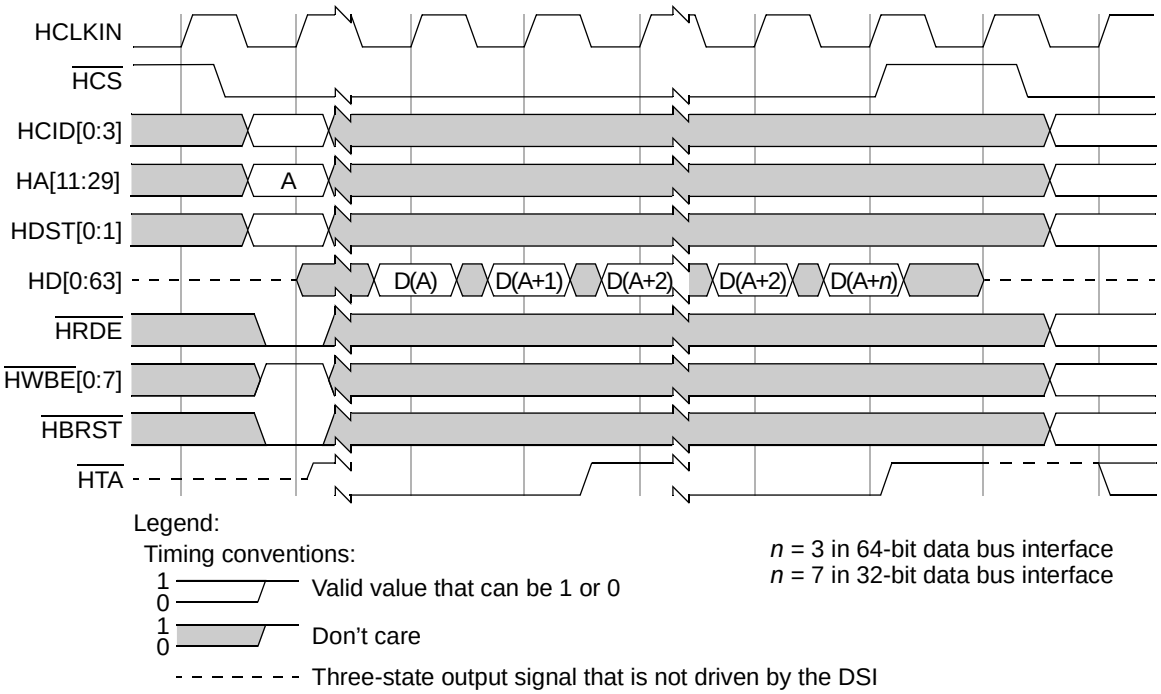


Figure 12-90. Synchronous Burst Read from MSC8102 DSI

$\overline{\text{HRDE}}$ and $\overline{\text{HBRST}}$ are asserted and $\overline{\text{HWBE}}$ are negated. When the DCR[8]:RPE bit (see the MSC8102 documentation) is set, a burst read access initiates data prefetching from consecutive addresses in the internal memory space. Assertion of $\overline{\text{HTA}}$ indicates that data is valid for the current beat of the access and the host must proceed to the next beat of this access. Because $\overline{\text{HTA}}$ is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until $\overline{\text{HTA}}$ goes to 0 and then the UPM continues in its pattern. When the host reaches the last beat of the access, it must terminate the burst access. The $\overline{\text{HTA}}$ is asserted earlier when the data for this access is already prefetched to the read buffer. Typically, after the first beat of the burst access, $\overline{\text{HTA}}$ remains asserted until the end of the access. After the last beat of the access, $\overline{\text{HTA}}$ is driven to 1 and stops being driven in the next rising edge of HCLKIN. The host can start its next access to the same MSC8102 immediately in the next HCLKIN rising edge without negating $\overline{\text{HCS}}$ between accesses. If the next access is not to the same MSC8102, to prevent contention on $\overline{\text{HTA}}$, the host must wait to access the next device until the previous DSI stops driving the $\overline{\text{HTA}}$ signal. The easiest way to achieve this is insert idle cycles at the end of the UPM pattern to guarantee that $\overline{\text{HTA}}$ is inactive.

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 between each DSI register access.

To avoid data corruption, if DER[0]:OVF is set, no broadcast access is written until the bit is reset. Therefore, after the last broadcast access, and before any regular write access, DER[0]:OVF must first be read and reset if it is set.

NOTE

In asynchronous mode, write data from a previous access (even a 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 two clock cycles (the DSI may translate accesses lasting longer than two 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 TMS320C*xxxx* 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 through 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 signals in Figure 12-48.

Table 12-48. 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, 0 MSB, 10 LSB, 11 Reserved	$\overline{\text{LBS}}$[0:1]
HCS	I	Chip select signal	$\overline{\text{LCS}}n$
HR/$\overline{\text{W}}$	I	R/$\overline{\text{W}}$ signal	LGPLy or inverted LBCTL
$\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$	I	Data strobe signals must be at least 2 DSP clocks wide • Host has separate active-low read and write strobe pins: connect one to $\overline{\text{HDS1}}$, one to $\overline{\text{HDS2}}$ • Host has one active-low strobe pin: connect to $\overline{\text{HDS1}}$ or $\overline{\text{HDS2}}$, the other to 1 • Host has one active high-strobe pin: connect to $\overline{\text{HDS1}}$ or $\overline{\text{HDS2}}$, the other to 0	LGPL*n*
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	INT*n*

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's two strobe pins, $\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$, allow different options to control transfers. The UPM supports 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 $\overline{\text{HDS}}$.

In any case, the host port requires a HR/$\overline{W}$ signal. This can be generated by using another LGPL signal and allows to adopt the timing virtually without restrictions. Alternatively the designer can invert LBCTL to generate this signal. It is the responsibility of the UPM pattern designer to plan for the additional delay of that inverter in the UPM pattern to satisfy AC timings at the DSP host port.

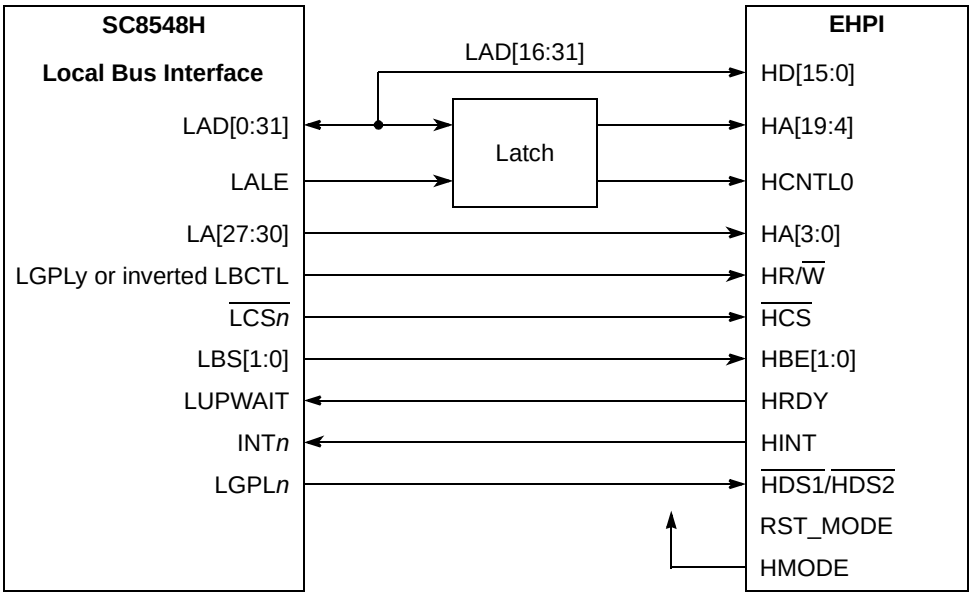


Figure 12-91. Interface to Texas Instruments EHPI in Non-Multiplexed Mode

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 LUPWAIT function of the UPM provides this mechanism. M*x*MR[UWPL] must be set to connect to this active-low signal.

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

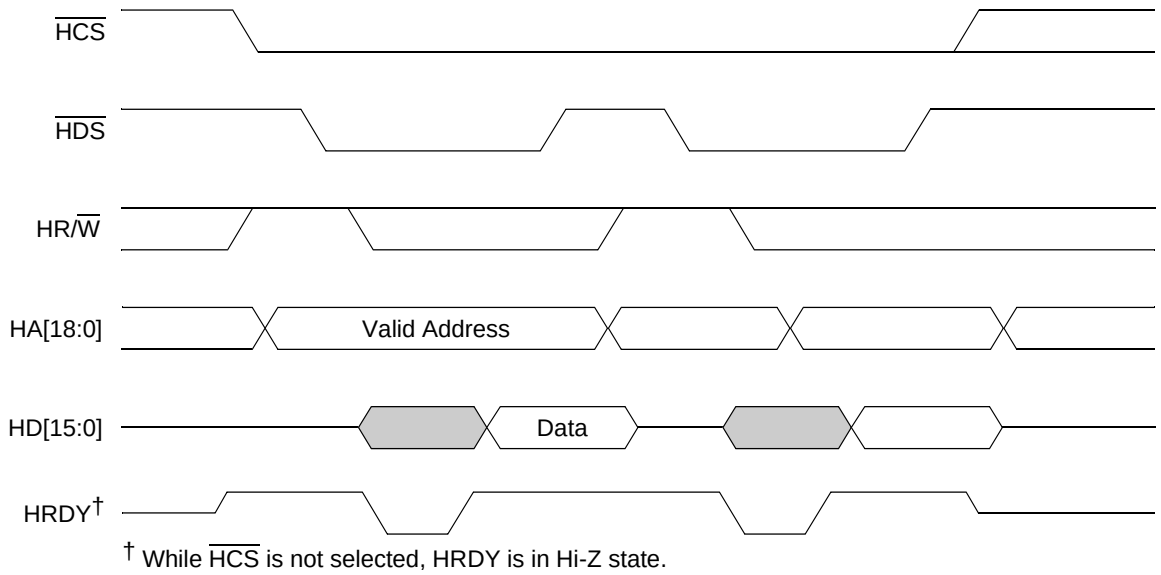


Figure 12-92. EHPI Non-Multiplexed Read Timings

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

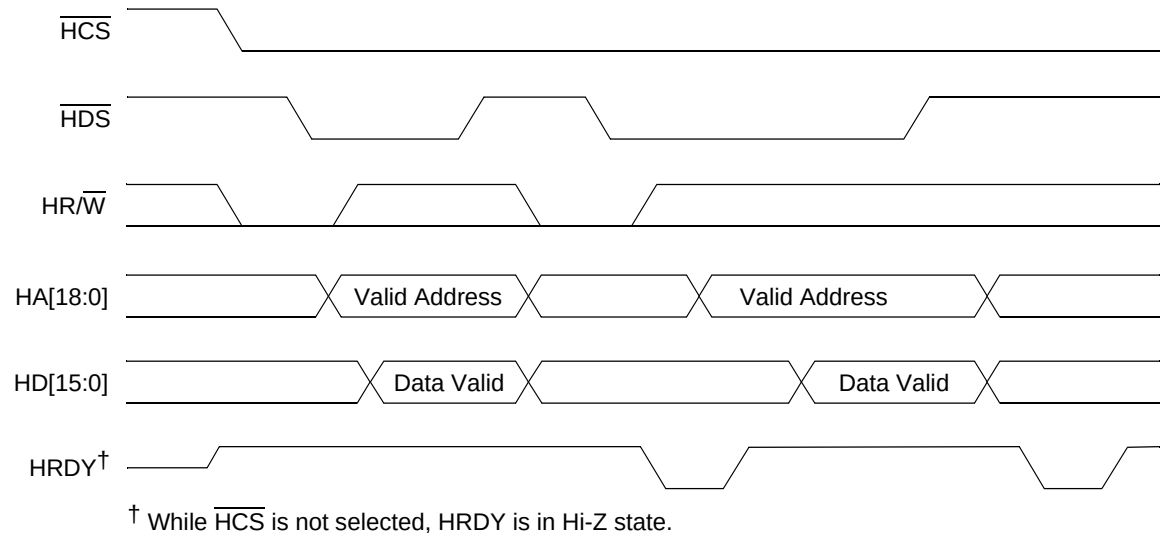


Figure 12-93. EHPI Non-Multiplexed Write Timings

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 for certain DSPs (such as, TMS320VC5510); for others (such as, TMS320VC5509), an external multiplexer must be used to decide which HRDY signal is routed to the local bus. This multiplexer 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 Controllers

This chapter describes the two three-speed Ethernet controllers of the MPC8560. The two controllers are referenced as TSEC1 and TSEC2.

13.1 Introduction

The Ethernet IEEE 802.3 protocol is widely used on LANs that are based on the carrier-sense multiple access/collision detect (CSMA/CD) approach. Because Ethernet and IEEE 802.3 protocols are similar and can coexist on the same LAN, both are referred to as Ethernet in this manual, unless otherwise noted. 10/100 Ethernet provides increased Ethernet speed from 10 to 100 megabits per second (Mbps) and provides a simple, cost-effective option for backbone and server connectivity. This three-speed Ethernet controller (TSEC) also implements a gigabit Ethernet protocol, which builds on top of the Ethernet protocol, but increases speed tenfold over 10/100 Ethernet to 1000 Mbps, or one gigabit per second (Gbps).

Gigabit Ethernet looks identical to Ethernet from the data link layer upward but it 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 layers. Figure 13-1 shows the typical Ethernet protocol stack and the relationship to the OSI model.

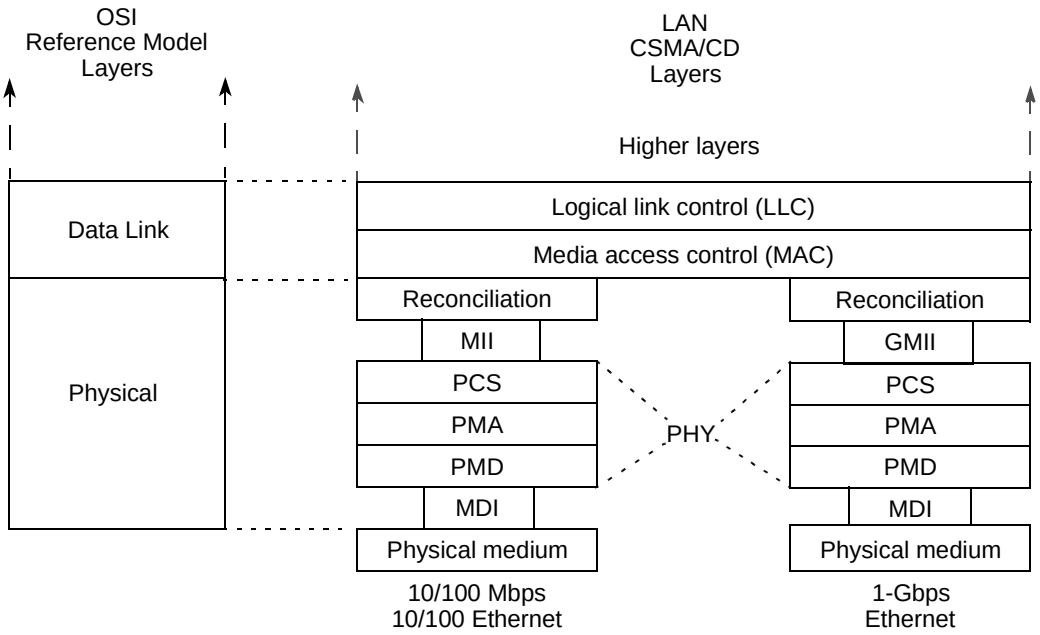


Figure 13-1. Ethernet Protocol in Relation to the OSI Protocol Stack

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 1-Gbps 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 encoding and decoding 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

(SerDes) and vice versa. Synchronization is also performed for proper data decoding in this sublayer. The PMA sits between the PCS and the PMD sublayers. For fiber medium (1000BASEX) the interface on the PMD side of the PMA is a 1-bit 1250-MHz signal, while on the PMA's PCS side the interface is a 10-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 PMD interface side of the PMA consists of four pair of 62.5-MHz PAM5 encoded signals, while the PCS 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.

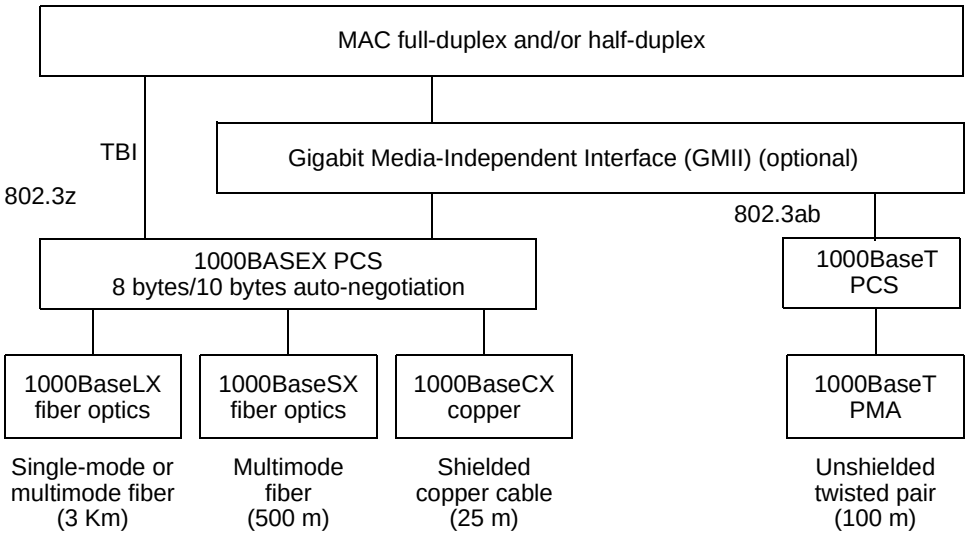


Figure 13-2. IEEE 802.3z and 802.3ab Physical Standards

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).

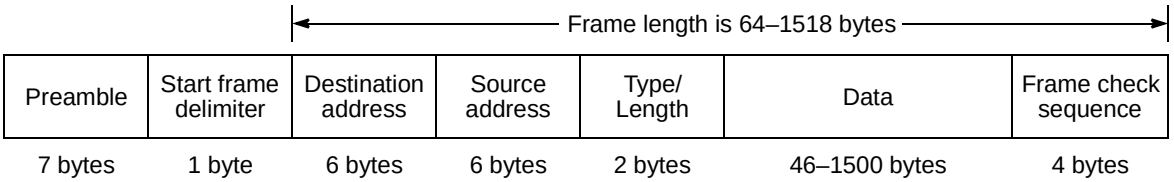


Figure 13-3. Ethernet/IEEE 802.3 Frame Structure

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 1,535 (0x5DD through 0x5FF) was intentionally left undefined.
- Data and padding—Padding is optional. It is only needed if the data is smaller than 46 octets (one octet = one byte) to ensure the minimum frame size of 64 octets as specified in the IEEE 802.3 standard. In 802.3x the first two octets of the data field are used as opcode (OP) (pause = 0x0001) and the second two octets are used to transmit a pause time (PT) parameter (pausetime = 0x0000 for on and 0xFFFF for off). In addition, a third two-octet field can be used for an extended pause control parameter (PTE). Because the use of these fields varies with the protocol used, the ability to examine them and report their content can significantly accelerate Ethernet frame processing.
- Frame-check sequence (FCS)—Specifies the standard 32-bit cyclic redundancy check (CRC) obtained using the standard CCITT-CRC polynomial on all fields except the preamble, SFD and CRC.

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

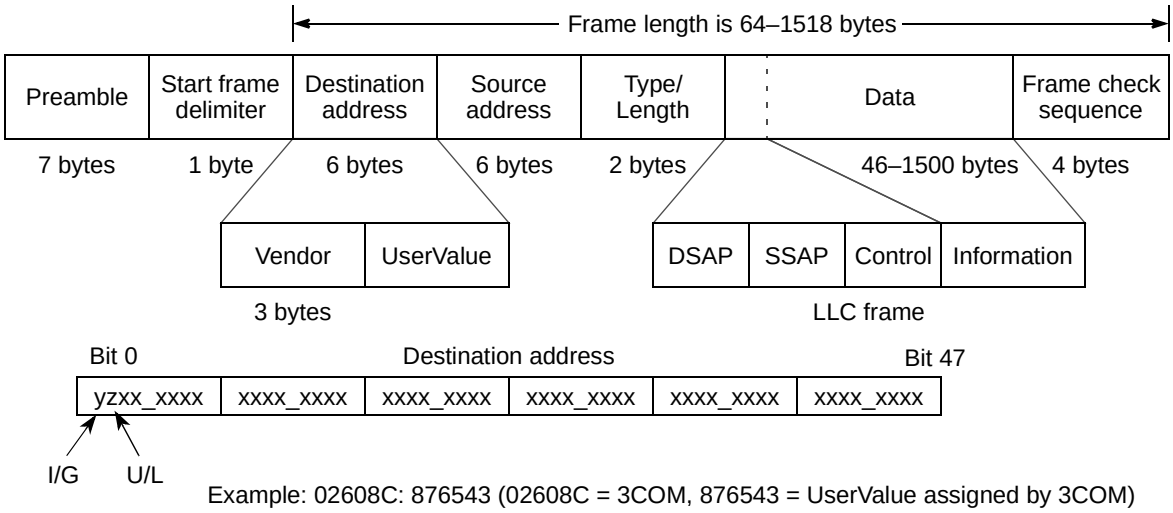


Figure 13-4. Ethernet/IEEE 802.3 Frame Structure With More Details

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

is transmitted bit by bit as

0100 0000 0000 0110 0011 0001 1110 0001 1010 0110 1100 0010

which is an individual address (because the lsb is cleared) but locally-defined (because the second least-significant bit is set).

When first originated, a type field was used for protocol identification. The IEEE 802.3 specification eliminated the type field, replacing it with the length field. The length field is used to identify the length, in bytes, of the data field. The protocol type in 802.3 frames are held within the data portion of the packet. The logical link control (LLC) is responsible for providing services to the network layer regardless of media type, such as FDDI, Ethernet, token ring, and others. The LLC layer makes use of LLC protocol data units (PDUs) in order to communicate between the media access control (MAC) layer and the upper layers of the protocol stack. Three variables determine access into the upper layers 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 = 0, 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 shows the block diagram of the TSEC.

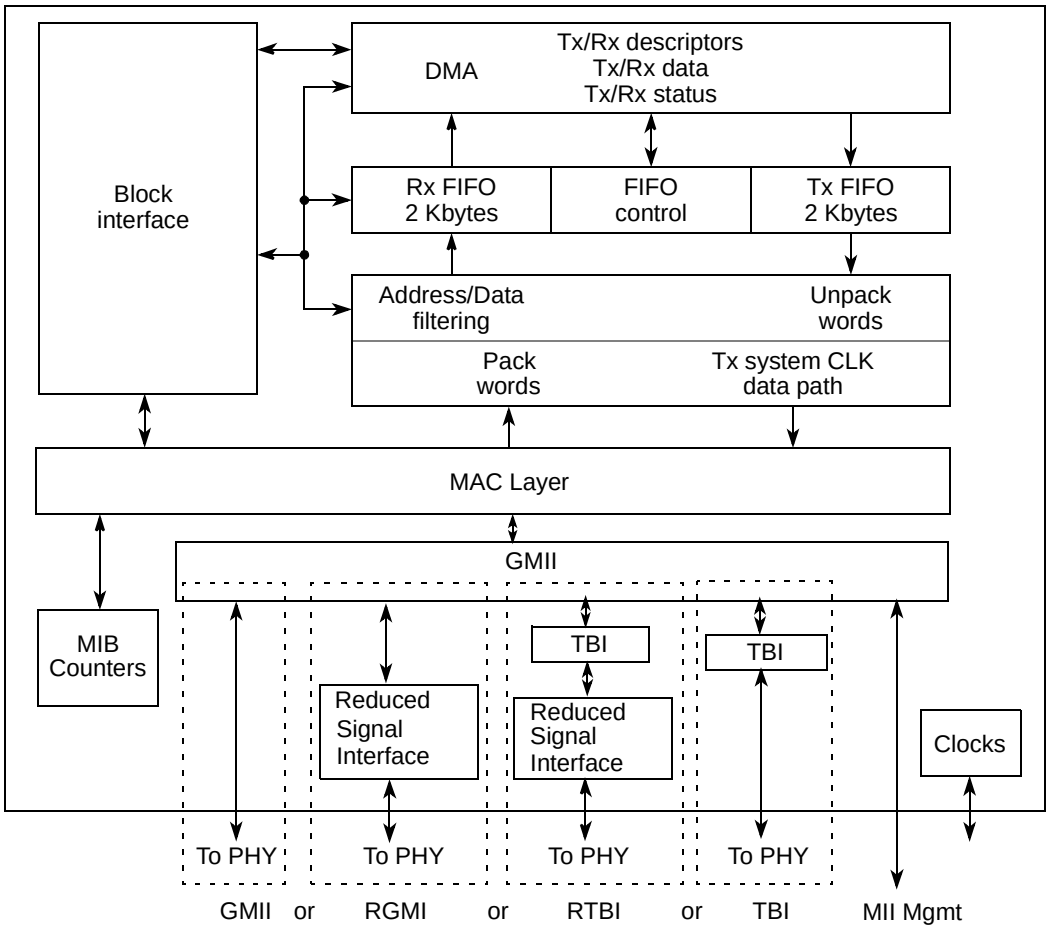


Figure 13-5. TSEC Block Diagram

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 1 Gbps
- TBI interface that can be connected to a SerDes device for fibre channel applications.
- Reduced signal count versions of the GMII (RGMII) and ten-bit (RTBI) interfaces

While most of this document refers to the non-reduced signal count interfaces, it must be understood that these references also apply to the reduced signal count interfaces.

The TSEC software programming model is similar to the MPC8260 (PowerQUICC II) device. Hence, it enables Freescale customers to leverage already implemented Ethernet drivers, reducing the software development cycle.

13.2 Features

The MPC8560 TSEC includes these distinctive features:

- IEEE 802.3, 802.3u, 802.3x, 802.3z, 802.3ac, 802.3ab compliant
- Support for different Ethernet physical interfaces:
  - 10/100/1Gb IEEE 802.3 GMII
  - 10/100 Mbps IEEE 802.3 MII
  - 10-Mbps IEEE 802.3 MII
  - 1-Gbps IEEE 802.3z TBI
  - 10/100 Mbps RMII
  - 1-Gbps full-duplex RGMII
  - 1-Gbps RTBI

- Full- and half-duplex support (1 Gbps 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 9.6 Kbytes) and IEEE 802.1 virtual local area network (VLAN) frames
- Retransmission from transmit FIFO following a collision
- Support for 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 the processor's L2 cache to reduce access latency
- Interrupt coalescing subject to a threshold frame counter and/or a threshold timer (Independent functionality for received and transmitted frames)
- RMON statistics support

13.3 Modes of Operation

The primary TSEC 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 (10/100/1Gb 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- and 100-Mbps MII interface operation

  The MAC-PHY interface operates in MII mode by configuring bits 22–23 of MACCFG2 (MACCFG2[I/F Mode] = 01). The MII is the media-independent interface defined by the 802.3 standard for 10/100 Mbps operation. The speed of operation is determined by the TSEC*n*_TX_CLK and TSEC*n*_RX_CLK signals, which are driven by the transceiver. The transceiver either auto-negotiates the speed or it may be controlled by software via the serial management interface (EC_MDC/EC_MDIO signals) to the transceiver.

- 1-Gbps GMII and TBI interface operation

  The MAC-PHY interface operates in GMII mode by configuring bits 22–23 of MACCFG2 (MACCFG2[I/F Mode] = 10). The GMII is the gigabit media-independent interface defined by the 802.3 standard for 1-Gbps 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 signal count mode (RGMII or RTBI), the MAC remains configured in GMII or TBI but the TSEC multiplexes and decodes the input signals 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 through the Loop Back bit in the MACCFG1 register. See Section 13.6.2.10, “Internal and External Loop Back,” for details.

13.4 External Signal Descriptions

This section defines the TSEC 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 signals (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 1-Gbps Ethernet rate.
- The TBI interface shares signals with the GMII interface signals.
- Finally, RGMII and RTBI options are reduced-signal implementations of the GMII and TBI interfaces.

13.4.1 Detailed Signal Descriptions

Table 13-1 contains detailed descriptions 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. Input signals not used are internally disabled. Output signals not used are driven low.

Table 13-1. TSEC Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TSEC*n*_COL	I	Collision input. The behavior of this signal is not specified while in full-duplex mode.	
		State Meaning	Asserted/Negated—In MII mode, this signal is asserted upon detection of a collision, and must remain asserted while the collision persists. This signal is not used in the TSEC GMII, TBI, RTBI and RGMII modes.
		Timing	Asserted/Negated—This signal is not required to transition synchronously with TSEC*n*_TX_CLK or TSEC*n*_RX_CLK.
TSEC*n*_CRS	I	Carrier sense input. In TBI and RTBI modes this signal can be used as SDET (signal detect), an optional signal that some PHYs generate in TBI or RTBI modes. This signal is not used in the TSEC GMII or RGMII modes.	
		State Meaning	Asserted/Negated—In MII 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 must remain asserted for the duration of the collision.
		Timing	Asserted/Negated—This signal is not required to transition synchronously with TSEC*n*_TX_CLK or TSEC*n*_RX_CLK.
TSEC*n*_GTX_CLK	O	Gigabit transmit clock. This signal is an output from the TSEC into the PHY. In GMII, TBI, or RTBI mode TSEC*n*_GTX_CLK is a 125-MHz clock that 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 not used in MII mode.	
EC_GTX_CLK125	I	Gigabit transmit 125-MHz source. This signal must be generated externally with a crystal or oscillator, or is sometimes provided by the PHY. 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 input is not used in MII mode.	

Table 13-1. TSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
EC_MDC	O	Management data clock. 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 a frequency of 2.5 MHz, but the device may be configured up to 12.5 MHz if supported by the PHY at that speed). This clock is generated by dividing the core complex bus (CCB) clock by eight. The ratio can be modified further by writing to MIIMCFG[29:31]. Note that this signal is used during reset to configure the TSEC interface to 'reduced-signal' or 'non-reduced-signal' mode. See Section 4.4.3, "Power-On Reset Configuration," for more information.
EC_MDIO	I/O	Management data input/output, BIDI
		State Meaning: Asserted/Negated—In GMII, MII, RGMII, RTBI or TBI mode EC_MDIO is a bidirectional signal to input PHY-supplied status during management read cycles and output control during MII management write cycles.
		Timing: Asserted/Negated—This signal is required to be synchronous with the EC_MDC signal.
TSEC*n*_RX_CLK	I	Receive clock. In GMII, MII or RGMII mode, the receive clock TSEC*n*_RX_CLK is a continuous clock (2.5, 25, or 125 MHz) that provides a timing reference for TSEC*n*_RX_DV, TSEC*n*_RXD, and TSEC*n*_RX_ER. In TBI mode, TSEC*n*_RX_CLK is the input for a 62.5-MHz PMA receive clock, 0 split phase with PMA_RX_CLK1 and is supplied by the SerDes. In RTBI mode it is a 125-MHz receive clock.
TSEC*n*_RX_DV	I	Receive data valid. In GMII or MII mode, if TSEC*n*_RX_DV is asserted, the PHY is indicating that valid data is present on the GMII and MII interfaces. In RGMII mode, TSEC*n*_RX_DV becomes RX_CTL. The RX_DV and RX_ERR are received on this signal on the rising and falling edges of TSEC*n*_RX_CLK. In TBI mode, TSEC*n*_RX_DV represents receive code group (RCG) bit 8. Together, with RCG[9] and RCG[7:0], they represents the 10-bit encoded symbol of GMII receive signals. In RTBI mode, TSEC*n*_RX_DV represents receive code group (RCG) bit 4 and 9. On the positive edge of the TSEC*n*_RX_CLK, RCG[4] and RCG[3:0] represents the first half of the 10-bit encoded symbol. On the negative edge of the TSEC*n*_RX_CLK, RCG[9] and RCG[8:4] represents the second half of the 10-bit encoded symbol.
TSEC*n*_RXD[7:0]	I	Receive data in. In GMII mode, TSEC*n*_RXD[7:4] with TSEC*n*_RXD[3:0], represent one complete octet of data to be transferred from the PHY to the MAC when TSEC*n*_RX_DV is asserted. In TBI mode, TSEC*n*_RXD[7:4] represents RCG[7:4]. Together, with RCG[9:8] and RCG[3:0], they represent the 10-bit encoded symbol of GMII receive signals. In GMII mode, TSEC*n*_RXD[3:0] represents a nibble of data to be transferred from the PHY to the MAC when TSEC*n*_RX_DV is asserted. A completely-formed SFD must be passed across the MII. While TSEC*n*_RX_DV is not asserted, TSEC*n*_RXD has no meaning. In RGMII 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.

Table 13-1. TSEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
TSEC*n*_RX_ER	I	Receive error
		State Meaning — Asserted/Negated—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. This signal is not used in the TSEC RTBI or RGMII modes.
TSEC*n*_TX_CLK	I	Transmit clock in. In MII mode, TSEC*n*_TX_CLK is a continuous clock (2.5 or 25 MHz) that provides a timing reference for the TSEC*n*_TX_EN, TSEC*n*_TXD, and TSEC*n*_TX_ER signals. In GMII mode, 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 and 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_CLK0, and is supplied by the SerDes. This signal is not used in the TSEC RTBI or RGMII modes.
TSEC*n*_TXD[7:0]	O	Transmit data out. 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. Note that some of these signals are also used during reset to configure the TSEC interface in GMII or TBI mode. Additionally, some of these signals are used for other reset configuration settings for the MPC8560. See Section 4.4.3, “Power-On Reset Configuration,” for more information.
TSEC*n*_TX_EN	O	Transmit data valid. In GMII or MII mode, if TSEC*n*_TX_EN is asserted, the MAC is indicating that valid data is present on the GMII’s or the MII’s TSEC*n*_TXD signals. In RGMII mode, TSEC*n*_TX_EN becomes TX_CTL. TX_EN and TX_ERR are asserted on this signal on rising and falling edges of the TSEC*n*_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. In RTBI mode, TSEC*n*_TX_EN represents TCG[4]. Together with TCG[9], TCG[3:0] and TCG[8:5], they represent the 10-bit encoded symbol.
TSEC*n*_TX_ER	O	Transmit error. In GMII or MII mode, assertion of TSEC*n*_TX_ER for one or more clock cycles while TSEC*n*_TX_EN is asserted causes the PHY to transmit one or more illegal symbols. Asserting TSEC*n*_TX_ER has no effect while operating at 10 Mbps or while TSEC*n*_TX_EN is negated. This signal transitions synchronously with respect to TSEC*n*_TX_CLK. In TBI mode, TSEC*n*_TX_ER represents TCG[9]. Together, with TCG[8:0], they represents the 10-bit encoded symbol. This signal is not used in the TSEC RTBI or RGMII modes.

13.5 Memory Map/Register Definition

The TSEC uses a software model similar to that employed by the Fast Ethernet function supported on the Freescale 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 access status information. The descriptors are used to pass data buffers and related buffer status or frame information between the hardware and software.

All accesses to and from the registers must be made with 32-bit accesses. There is no support for accesses of sizes other than 32 bits.

This section of the document defines the memory map and describes the registers in detail. The buffer descriptor is described in Section 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	TSEC MAC registers

Table 13-2. Module Memory Map Summary (continued)

600–7FF	TSEC RMON MIB registers
800–8FF	TSEC HASH function registers
900–AFF	Reserved
B00–BFF	TSEC attribute registers
C00–FFF	Future expansion space

13.5.2 Detailed Memory Map—Control/Status Registers

Table 13-3 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 13-3, but the registers for TSEC2 are not. Note in Table 13-3 that the registers are the same for TSEC2 except that the offset changes from 0x2_4*nnn* to 0x2_5*nnn*.

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	IEVENT—Interrupt event register	R/W	0x0000_0000	13.5.3.1.1/13-20
0x2_4014	IMASK—Interrupt mask register	R/W	0x0000_0000	13.5.3.1.2/13-23
0x2_4018	EDIS—Error disabled register	R/W	0x0000_0000	13.5.3.1.3/13-24
0x2_401C	Reserved	R	0x0000_0000	—
0x2_4020	ECNTRL—Ethernet control register	R/W	0x0000_0000	13.5.3.1.4/13-25
0x2_4024	MINFLR—Minimum frame length register	R/W	0x0000_0040	13.5.3.1.5/13-26
0x2_4028	PTV—Pause time value register	R/W	0x0000_0000	13.5.3.1.6/13-27
0x2_402C	DMACTRL—DMA control register	R/W	0x0000_0000	13.5.3.1.7/13-28
0x2_4030	TBIPA—TBI PHY address register	R/W	0x0000_0000	13.5.3.1.8/13-29
0x2_4034–0x2_4048	Reserved	R	0x0000_0000	—
TSEC1 FIFO Control and Status Registers				
0x2_404C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	13.5.3.2.1/13-30
0x2_4050–0x2_4088	Reserved	R	0x0000_0000	—
0x2_408C	FIFO_TX_THR—FIFO transmit threshold register	R/W	0x0000_0100	13.5.3.2.2/13-31
0x2_4090–0x2_4094	Reserved	R	0x0000_0000	—

Table 13-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_4098	FIFO_TX_STARVE—FIFO transmit starve register	R/W	0x0000_0080	13.5.3.2.3/13-32
0x2_409C	FIFO_TX_STARVE_SHUTOFF—FIFO transmit starve shutoff register	R/W	0x0000_0100	13.5.3.2.4/13-32
0x2_40A0–0x2_40FC	Reserved	R	0x0000_0000	—
TSEC1 Transmit Control and Status Registers				
0x2_4100	TCTRL—Transmit control register	R/W	0x0000_0000	13.5.3.3.1/13-33
0x2_4104	TSTAT—Transmit status register	R/W	0x0000_0000	13.5.3.3.2/13-34
0x2_4108	Reserved	R	0x0000_0000	—
0x2_410C	TBDLEN—TxBD data length register	R	0x0000_0000	13.5.3.3.3/13-35
0x2_4110	TXIC—Transmit interrupt coalescing configuration register	R/W	0x0000_0000	13.5.3.3.4/13-35
0x2_4114–0x2_4120	Reserved	R	0x0000_0000	—
0x2_4124	CTBPTR—Current TxBD pointer register	R	0x0000_0000	13.5.3.3.5/13-36
0x2_4128–0x2_4180	Reserved	R	0x0000_0000	—
0x2_4184	TBPTR—TxBD pointer register	R/W	0x0000_0000	13.5.3.3.6/13-37
0x2_4188–0x2_4200	Reserved	R	0x0000_0000	—
0x2_4204	TBASE—TxBD base address register	R/W	0x0000_0000	13.5.3.3.7/13-37
0x2_4208–0x2_42AC	Reserved	R	0x0000_0000	—
0x2_42B0	OSTBD—Out-of-sequence TxBD register	R/W	0x0800_0000	13.5.3.3.8/13-38
0x2_42B4	OSTBDP—Out-of-sequence Tx data buffer pointer register	R/W	0x0000_0000	13.5.3.3.9/13-40
0x2_42B8–0x2_42FC	Reserved	R	0x0000_0000	—
TSEC1 Receive Control and Status Registers				
0x2_4300	RCTRL—Receive control register	R/W	0x0000_0000	13.5.3.4.1/13-41
0x2_4304	RSTAT—Receive status register	R/W	0x0000_0000	13.5.3.4.2/13-41
0x2_4308	Reserved	R	0x0000_0000	—
0x2_430C	RBDLEN—RxBD data length register	R	0x0000_0000	13.5.3.4.3/13-42
0x2_4310	RXIC—Receive interrupt coalescing configuration register	R/W	0x0000_0000	13.5.3.4.4/13-43
0x2_4314–0x2_4320	Reserved	R	0x0000_0000	—

Table 13-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_4324	CRBPTR—Current RxBD pointer register	R	0x0000_0000	13.5.3.4.5/13-44
0x2_4328–0x2_433C	Reserved	R	0x0000_0000	—
0x2_4340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	13.5.3.4.6/13-44
0x2_4344–0x2_4380	Reserved	R	0x0000_0000	—
0x2_4384	RBPTR—RxBD pointer register	R/W	0x0000_0000	13.5.3.4.7/13-45
0x2_4388–0x2_4400	Reserved	R	0x0000_0000	—
0x2_4404	RBASE—RxBD base address register	R/W	0x0000_0000	13.5.3.4.8/13-45
0x2_4408–0x2_44FC	Reserved	R	0x0000_0000	—
TSEC1 MAC Registers				
0x2_4500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	13.5.3.6.1/13-49
0x2_4504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	13.5.3.6.2/13-51
0x2_4508	IPGIFG—Inter-packet gap/inter-frame gap register	R/W	0x4060_5060	13.5.3.6.3/13-52
0x2_450C	HAFDUP—Half-duplex register	R/W	0x00A1_F037	13.5.3.6.4/13-53
0x2_4510	MAXFRM—Maximum frame length register	R/W	0x0000_0600	13.5.3.6.5/13-54
0x2_4514–0x2_451C	Reserved	R	0x0000_0000	—
0x2_4520	MIIMCFG—MII management configuration register	R/W	0x0000_0000	13.5.3.6.6/13-54
0x2_4524	MIIMCOM—MII Management command register	R/W	0x0000_0000	13.5.3.6.7/13-55
0x2_4528	MIIMADD—MII Management address register	R/W	0x0000_0000	13.5.3.6.8/13-56
0x2_452C	MIIMCON—MII Management control register	W	0x0000_0000	13.5.3.6.9/13-57
0x2_4530	MIIMSTAT—MII Management status register	R	0x0000_0000	13.5.3.6.10/13-57
0x2_4534	MIIMIND—MII Management indicator register	R	0x0000_0000	13.5.3.6.11/13-58
0x2_4538	Reserved	R	0x0000_0000	—
0x2_453C	IFSTAT—Interface status register	R/W	0x0000_0000	13.5.3.6.12/13-58
0x2_4540	MACSTNADDR1—Station address register, part 1	R/W	0x0000_0000	13.5.3.6.13/13-59

Table 13-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_4544	MACSTNADDR2—Station address register, part 2	R/W	0x0000_0000	13.5.3.6.14/13-60
0x2_4548–0x2_467C	Reserved	R	0x0000_0000	—
TSEC1 RMON MIB Registers				
TSEC1 Transmit and Receive Counters				
0x2_4680	TR64—Transmit and receive 64-byte frame counter register	R/W	0x0000_0000	13.5.3.7.1/13-61
0x2_4684	TR127—Transmit and receive 65- to 127-byte frame counter register	R/W	0x0000_0000	13.5.3.7.2/13-61
0x2_4688	TR255—Transmit and receive 128- to 255-byte frame counter register	R/W	0x0000_0000	13.5.3.7.3/13-62
0x2_468C	TR511—Transmit and receive 256- to 511-byte frame counter register	R/W	0x0000_0000	13.5.3.7.4/13-62
0x2_4690	TR1K—Transmit and receive 512- to 1023-byte frame counter register	R/W	0x0000_0000	13.5.3.7.5/13-63
0x2_4694	TRMAX—Transmit and receive 1024- to 1518-byte frame counter register	R/W	0x0000_0000	13.5.3.7.6/13-63
0x2_4698	TRMGV—Transmit and receive 1519- to 1522-byte good VLAN frame count register	R/W	0x0000_0000	13.5.3.7.7/13-64
TSEC1 Receive Counters				
0x2_469C	RBYT—Receive byte counter register	R/W	0x0000_0000	13.5.3.7.8/13-64
0x2_46A0	RPKT—Receive packet counter register	R/W	0x0000_0000	13.5.3.7.9/13-65
0x2_46A4	RFCS—Receive FCS error counter register	R/W	0x0000_0000	13.5.3.7.10/13-65
0x2_46A8	RMCA—Receive multicast packet counter register	R/W	0x0000_0000	13.5.3.7.11/13-66
0x2_46AC	RBCA—Receive broadcast packet counter register	R/W	0x0000_0000	13.5.3.7.12/13-66
0x2_46B0	RXCF—Receive control frame packet counter register	R/W	0x0000_0000	13.5.3.7.13/13-67
0x2_46B4	RXPF—Receive PAUSE frame packet counter register	R/W	0x0000_0000	13.5.3.7.14/13-67
0x2_46B8	RXUO—Receive unknown OP code counter register	R/W	0x0000_0000	13.5.3.7.15/13-68
0x2_46BC	RALN—Receive alignment error counter register	R/W	0x0000_0000	13.5.3.7.16/13-68
0x2_46C0	RFLR—Receive frame length error counter register	R/W	0x0000_0000	13.5.3.7.17/13-69
0x2_46C4	RCDE—Receive code error counter register	R/W	0x0000_0000	13.5.3.7.18/13-69
0x2_46C8	RCSE—Receive carrier sense error counter register	R/W	0x0000_0000	13.5.3.7.19/13-70
0x2_46CC	RUND—Receive undersize packet counter register	R/W	0x0000_0000	13.5.3.7.20/13-70
0x2_46D0	ROVR—Receive oversize packet counter register	R/W	0x0000_0000	13.5.3.7.21/13-71
0x2_46D4	RFRG—Receive fragments counter register	R/W	0x0000_0000	13.5.3.7.22/13-71
0x2_46D8	RJBR—Receive jabber counter register	R/W	0x0000_0000	13.5.3.7.23/13-72

Table 13-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_46DC	RDRP—Receive drop register	R/W	0x0000_0000	13.5.3.7.24/13-72
TSEC1 Transmit Counters				
0x2_46E0	TBYT—Transmit byte counter register	R/W	0x0000_0000	13.5.3.7.25/13-73
0x2_46E4	TPKT—Transmit packet counter register	R/W	0x0000_0000	13.5.3.7.26/13-73
0x2_46E8	TMCA—Transmit multicast packet counter register	R/W	0x0000_0000	13.5.3.7.27/13-74
0x2_46EC	TBCA—Transmit broadcast packet counter register	R/W	0x0000_0000	13.5.3.7.28/13-74
0x2_46F0	TXPF—Transmit PAUSE control frame counter register	R/W	0x0000_0000	13.5.3.7.29/13-75
0x2_46F4	TDFR—Transmit deferral packet counter register	R/W	0x0000_0000	13.5.3.7.30/13-75
0x2_46F8	TEDF—Transmit excessive deferral packet counter register	R/W	0x0000_0000	13.5.3.7.31/13-76
0x2_46FC	TSCL—Transmit single collision packet counter register	R/W	0x0000_0000	13.5.3.7.32/13-76
0x2_4700	TMCL—Transmit multiple collision packet counter register	R/W	0x0000_0000	13.5.3.7.33/13-77
0x2_4704	TLCL—Transmit late collision packet counter register	R/W	0x0000_0000	13.5.3.7.34/13-77
0x2_4708	TXCL—Transmit excessive collision packet counter register	R/W	0x0000_0000	13.5.3.7.35/13-78
0x2_470C	TNCL—Transmit total collision counter register	R/W	0x0000_0000	13.5.3.7.36/13-78
0x2_4710	Reserved	R	0x0000_0000	—
0x2_4714	TDRP—Transmit drop frame counter register	R/W	0x0000_0000	13.5.3.7.37/13-79
0x2_4718	TJBR—Transmit jabber frame counter register	R/W	0x0000_0000	13.5.3.7.38/13-79
0x2_471C	TFCS—Transmit FCS error counter register	R/W	0x0000_0000	13.5.3.7.39/13-80
0x2_4720	TXCF—Transmit control frame counter register	R/W	0x0000_0000	13.5.3.7.40/13-80
0x2_4724	TOVR—Transmit oversize frame counter register	R/W	0x0000_0000	13.5.3.7.41/13-81
0x2_4728	TUND—Transmit undersize frame counter register	R/W	0x0000_0000	13.5.3.7.42/13-81
0x2_472C	TFRG—Transmit fragments frame counter register	R/W	0x0000_0000	13.5.3.7.43/13-82
TSEC1 General Registers				
0x2_4730	CAR1—Carry register one	R/W	0x0000_0000	13.5.3.7.44/13-82
0x2_4734	CAR2—Carry register two	R/W	0x0000_0000	13.5.3.7.45/13-84
0x2_4738	CAM1—Carry register one mask register	R/W	0xFE01_FFFF	13.5.3.7.46/13-85
0x2_473C	CAM2—Carry register two mask register	R/W	0x000F_FFFF	13.5.3.7.47/13-86
0x2_4740–0x2_47FC	Reserved	R	0x0000_0000	—

Table 13-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
TSEC1 Hash Function Registers				
0x2_4800	IADDR0—Individual address register 0	R/W	0x0000_0000	13.5.3.8.1/13-88
0x2_4804	IADDR1—Individual address register 1	R/W	0x0000_0000	
0x2_4808	IADDR2—Individual address register 2	R/W	0x0000_0000	
0x2_480C	IADDR3—Individual address register 3	R/W	0x0000_0000	
0x2_4810	IADDR4—Individual address register 4	R/W	0x0000_0000	
0x2_4814	IADDR5—Individual address register 5	R/W	0x0000_0000	
0x2_4818	IADDR6—Individual address register 6	R/W	0x0000_0000	
0x2_481C	IADDR7—Individual address register 7	R/W	0x0000_0000	
0x2_4820–0x2_487C	Reserved	R	0x0000_0000	—
0x2_4880	GADDR0—Group address register 0	R/W	0x0000_0000	13.5.3.8.2/13-88
0x2_4884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x2_4888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x2_488C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x2_4890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x2_4894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x2_4898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x2_489C	GADDR7—Group address register 7	R/W	0x0000_0000	
0x2_48A0–0x2_4AFC	Reserved	R	0x0000_0000	—
TSEC1 Attribute Registers				
0x2_4B00–0x2_4BF4	Reserved	R	0x0000_0000	—
0x2_4BF8	ATTR—Attribute register	R	0x0000_0000	13.5.3.9.1/13-89
0x2_4BFC	ATTRELI—Attribute EL & EI register	R	0x0000_0000	13.5.3.9.2/13-90
TSEC1 Future Expansion Space				
0x2_4C00–0x2_4FFC	Reserved	R	0x0000_0000	—
TSEC2 Registers				
0x2_5000–0x2_5FFC	TSEC2 registers[2]			

1 R = means read-only, W = write only, R/W = read and write, LH = latches high, SC = self-clearing.

2 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, shown in Figure 13-6, an interrupt is generated if the corresponding bit in the interrupt enable register (IMASK) is also set. Clearing the IEVENT bit clears the interrupt signal. The bit in the IEVENT register is cleared if a 1 is written to that bit position. A write of 0 has no effect.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts that may occur in normal operation are:

- GTSC, GRSC, TXF, TXB, TXC, RXF, RXB, RXC, and MSRO

Interrupts resulting from errors/problems detected in the network or transceiver are:

- BABR, BABT, LC, and CRL/XDA

Interrupts resulting from internal errors are:

- EBERR, XFUN, and BSY

Some of the error interrupts are independently counted in the management information base (MIB) block counters. Software may choose to mask off these interrupts because these errors are visible to network management through the MIB counters.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BABR	RXC	BSY	EBERR	0	MSRO	GTSC	BABT	TXC	TXE	TXB	TXF	0	LC	CRL/ XDA	XFUN
W																
Reset	0000_0000_0000_0000															

	16	17					22	23	24	25						31
R	RXB	0	0	0	0	0	0	GRSC	RXF	0	0	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4010; TSEC2:0x2_5010															

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[Huge Frame] is set. 0 Excessive frame not received 1 Excessive frame received
1	RXC	Receive control interrupt. A control frame was received. If MACCFG1[Rx_Flow] is set, a pause operation is performed lasting for the duration specified in the received pause control frame and beginning when the frame was received. 0 Control frame not received 1 Control frame received
2	BSY	Busy condition interrupt. Indicates that a frame was received and discarded due to a lack of buffers. When IEVENT[BSY] is set RSTAT[QHLT] is also set. In order to begin receiving packets again, the user must clear RSTAT[QHLT]. This bit and RSTAT[QHLT] are set whenever the TSEC reads an RxBD with its Empty field cleared. 0 No frame received and discarded 1 Frame received and discarded
3	EBERR	Ethernet bus error. This bit indicates that a system bus error for a memory read occurred while a DMA transaction was underway. If the EBERR is set while transmission is in progress, the DMA stops sending data to the Tx FIFO which eventually causes an underrun error (XFUN) and TSTAT[THLT] is set. If the EBERR is set while receiving a frame, the DMA discards the frame and RSTAT[QHLT] is set. 0 No system bus error occurred 1 System bus error occurred
4	—	Reserved
5	MSRO	MSTAT Register Overflow. This interrupt is asserted if the count for one of the MSTAT registers has exceeded the size of the register. 0 MSTAT count not exceeding register size 1 MSTAT count exceeds register size
6	GTSC	Graceful transmit stop complete. This interrupt is asserted following the completion of a graceful stop, which was intitiated by setting either DMACTRL[GTS] or TCTRL[TFC_PAUSE]. Graceful stop means that the transmitter is put into a pause state after completion of the frame currently being transmitted. 0 Graceful transmit stop not complete or not initiated 1 Graceful transmit stop completed
7	BABT	Babbling transmit error. This bit indicates that the transmitted frame length has exceeded the value in the MAC's Maximum Frame Length register and MACCFG2[Huge Frame] is cleared. Frame truncation occurs when this condition occurs. TxBD[TXTRUNC] is set in the last TxBD (TxBD[L] is set) of the frame. 0 Transmitted frame length not exceeding maximum frame length 1 Transmitted frame length exceeding maximum frame length
8	TXC	Transmit control interrupt. This bit indicates that a control frame was transmitted. 0 Control frame not transmitted 1 Control frame transmitted

Table 13-4. IEVENT Field Descriptions (continued)

Bits	Name	Description
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/XDA, XFUN). It is not set if DMACTRL[WOP] is set and TSEC runs out of TxBDs to process. In order to begin transmitting packets again, the user must clear TSTAT[THLT]. 0 No transmit channel error occurred 1 Transmit channel error occurred
10	TXB	Transmit buffer. This bit indicates that a transmit buffer descriptor was updated whose I (interrupt) bit was set in its status word and was not the last buffer descriptor of the frame. 0 No transmit buffer descriptor updated 1 Transmit buffer descriptor updated
11	TXF	Transmit frame interrupt. This bit indicates that a frame was transmitted and that the last corresponding transmit buffer descriptor (TxBD) was updated. This only occurs if the I (interrupt) bit in the status word of the buffer descriptor is set. 0 No frame transmitted/TxBD not updated 1 Frame transmitted/TxBD updated
12	—	Reserved
13	LC	Late collision. This bit indicates that a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame is truncated with a bad CRC and the remainder of the frame is discarded. 0 No late collision occurred 1 Late collision occurred
14	CRL/ XDA	Collision retry limit/excessive defer abort. This bit indicates either one of two conditions occurred while attempting to transmit a frame: 1) the number of successive transmission collisions has exceeded the MAC's HAFDUP[Retransmission Maximum] count or 2) an excessive defer abort condition has occurred. An excessive defer abort condition occurs when the TSEC waits more than 3036 bytes while attempting to send a frame and HAFDUP[EXCESS_DEFER] is 0. The TxBD[DEF] or OSTBD[DEF] is also set. In either case the frame is discarded without being transmitted and the TSEC is halted (TSTAT[THLT] is set). The CRL or XDA condition can only occur while in half-duplex mode. 0 Successive transmission collisions do not exceed maximum and no excessive defer abort condition has occurred. 1 Successive transmission collisions exceed maximum or an excessive defer abort condition has occurred.
15	XFUN	Transmit FIFO underrun. This bit indicates that the transmit FIFO became empty before the complete frame was transmitted. 0 Transmit FIFO not underrun 1 Transmit FIFO underrun
16	RXB	Receive buffer. These bits indicate that a receive buffer descriptor was updated which had the I (Interrupt) bit set in its status word and was not the last buffer descriptor of the frame. 0 Receive buffer descriptor not updated 1 Receiver buffer descriptor updated
17–22	—	Reserved
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

Table 13-4. IEVENT Field Descriptions (continued)

Bits	Name	Description
24	RXF	Receive frame interrupt. The last receive buffer descriptor (RxBD) of a frame was updated. This occurs only if the I (interrupt) bit in the buffer descriptor status word is set. 0 Frame not received 1 Frame received
25–31	—	Reserved

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 can be cleared by clearing the corresponding IEVENT bit.

Figure 13-7 shows 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	CRL/ XDAEN	XFUNEN
W																
Reset	0000_0000_0000_0000															

	16	17				22	23	24	25							31
R	RXBEN	0	0	0	0	0	0	GRSCEN	RXFEN	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
2	BSYEN	Busy interrupt enable
3	EBERREN	Ethernet controller bus error enable
4	—	Reserved
5	MSROEN	MSTAT register overflow interrupt enable

Table 13-5. IMASK Field Descriptions (continued)

Bits	Name	Description
6	GTSCEN	Graceful transmit stop complete interrupt enable
7	BTEN	Babbling transmitter interrupt enable
8	TXCEN	Transmit control interrupt enable
9	TXEEN	Transmit error interrupt enable
10	TXBEN	Transmit buffer interrupt enable
11	TXFEN	Transmit frame interrupt enable
12	—	Reserved
13	LCEN	Late collision enable
14	CRL/XDAEN	Collision retry limit/excessive defer enable
15	XFUNEN	Transmit FIFO underrun enable
16	RXBEN	Receive buffer interrupt enable
17–22	—	Reserved
23	GRSCEN	Graceful receive stop complete interrupt enable
24	RXFEN	Receive frame interrupt enable
25–31	—	Reserved

13.5.3.1.3 Error Disabled Register (EDIS)

The error disabled register, shown in Figure 13-8, controls error reporting by the TSEC. The IEVENT bit corresponding to an error will not be set if the error is disabled in EDIS.

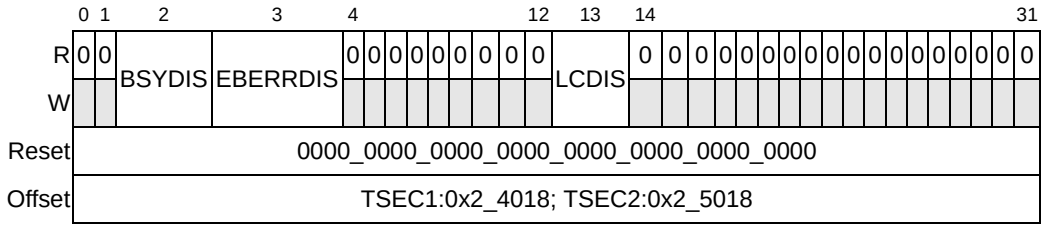


Figure 13-8. Error Disabled Register (EDIS)

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.
4–12	—	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.
14–31	—	Reserved

13.5.3.1.4 Ethernet Control Register (ECNTRL)

ECNTRL, shown in Figure 13-9, is used to reset, configure, and initialize the TSEC.

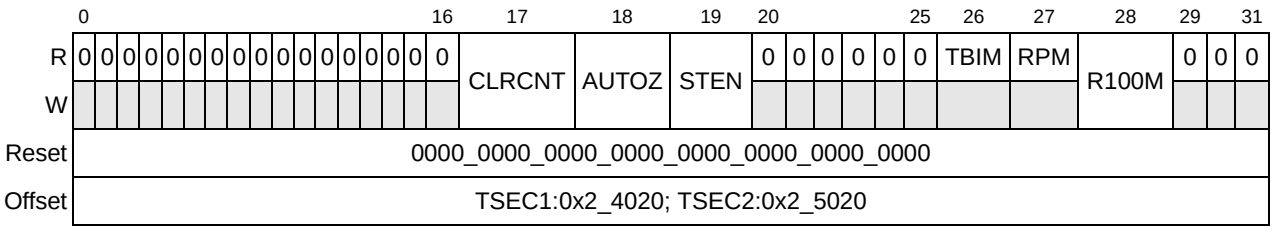


Figure 13-9. ECNTRL Register Definition

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.

Table 13-7. ECNTRL Field Descriptions (continued)

Bits	Name	Description
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 signal configured at reset to set or clear using the TSEC*n*_GTX_CLK signal. 0 GMII or MII mode interface 1 TBI mode interface
27	RPM	Reduced signal mode. If this bit is set, a reduced signal 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 signal configured at reset to 0 or 1 with the EC_MDC signal. 0 GMII or TBI in non-reduced signal mode configuration 1 RGMII or RTBI reduced signal 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 shows the MINFLR register.

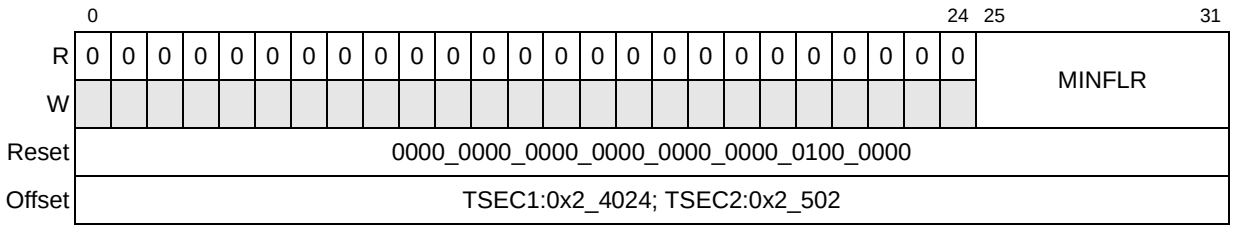


Figure 13-10. MINFLR Register Definition

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, shown in Figure 13-11, is 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 pause time and the high-order 16 bits (PTE) represent the extended pause control parameter. The pause time is measured in units of pause_quanta, equal to 512 bit times. The pause time can range from 0 to 65,535 pause_quanta, or 0 to 33,553,920 bit times. See Section 13.6.2.7, “Flow Control for additional details.

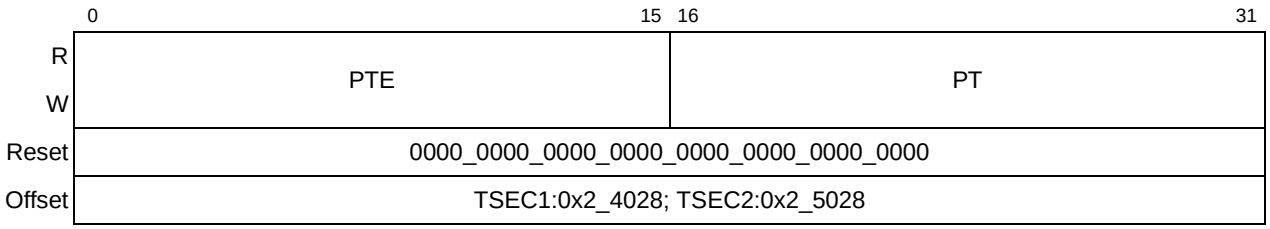


Figure 13-11. PTV Register Definition

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

Table 13-9. PTV Field Descriptions

Bits	Name	Description
0–15	PTE	Extended pause control. This field allows software to add a 16-bit additional control parameter into the pause frame to be sent when TCTRL[TFC_PAUSE] is set. Note that current IEEE 802.3 pause frame format requires this parameter to be set to 0.
16–31	PT	Pause time value. Represents the 16-bit pause quanta (that is, 512 bit times). This pause value is used as part of the pause frame to be sent when TCTRL[TFC_PAUSE] is set. See Section 13.6.2.7, “Flow Control,” for more information.

13.5.3.1.7 DMA Control Register (DMACTRL)

DMACTRL, shown in Figure 13-12, is used to configure the DMA block. register.

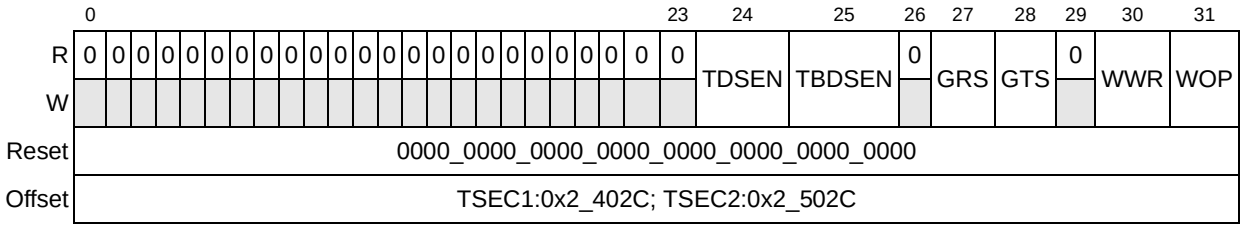


Figure 13-12. DMACTRL Register Definition

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 MAC's receive enable bit (MACCFG[Rx_EN]) 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	—	Reserved

Table 13-10. DMACTRL Field Descriptions (continued)

Bits	Name	Description
30	WWR	Write with response. This bit gives the user the assurance that a BD was updated in memory before it receives an interrupt concerning a transmit or receive frame. 0 Do not wait for acknowledgement from system for BD writes before setting IEVENT bits. 1 Before setting IEVENT bits TXB, TXF, TXE, IE, XFUN, LC, CRL/XDA, RXB, RXF, the 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. (IEVENT[TXE] is clear.) To resume transmission, software must clear TSTAT[THLT]. Note that if this bit is set, the user must ensure that all TxBDs involved with sending a frame have their ready bits set before any transmission begins. Otherwise, TSEC behaves in a boundedly undefined fashion. A buffer descriptor is considered not ready when its TxBD[Ready] field is clear or its TxBD[Data Length] field is zero. It is considered ready when both TxBD[Ready] is set and TxBD[Data Length] is non-zero. In Wait mode (DMACTRL[WOP] is set) when TSEC is processing a frame and an intermediate TxBD’s ready bit is cleared, TSEC does not halt, but instead continuously polls the same TxBD until the TxBD becomes ready or an Ethernet interface error or a memory error is encountered. CxBD becomes ready, TSEC reports the error in both the IEVENT register as well as the TxBD for Ethernet interface errors, or the IEVENT[EBERR] for a memory error and sets the TSTAT[THLT] bit. Note that software must eventually set all of its TxBDs for a frame, because TSEC continuously reads an intermediate TxBD until it becomes ready if insufficient data has been read to surpass the FIFO Transmit Threshold (FIFO_TX_THR) register value. 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 TBIPA, shown in Figure 13-13, 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).”

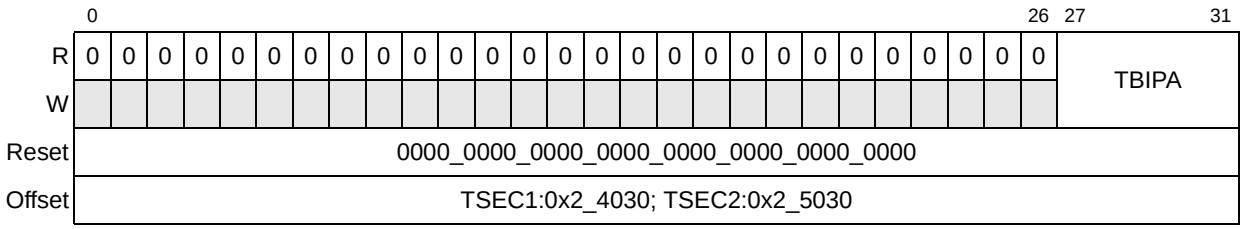


Figure 13-13. TBIPA Register Definition

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

Table 13-11. TBIPA Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27–31	TBIPA	TBI PHY address. 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 bytes) is used to indicate that the amount of data in the FIFO is so low that the risk of underrun is extremely high. The FIFO_TX_STARVE_SHUTOFF (default is 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 Pause Control Register (FIFO_PAUSE_CTRL)

FIFO_PAUSE_CTRL, shown in Figure 13-14, is writable by the user to configure the properties of the TSEC FIFO.

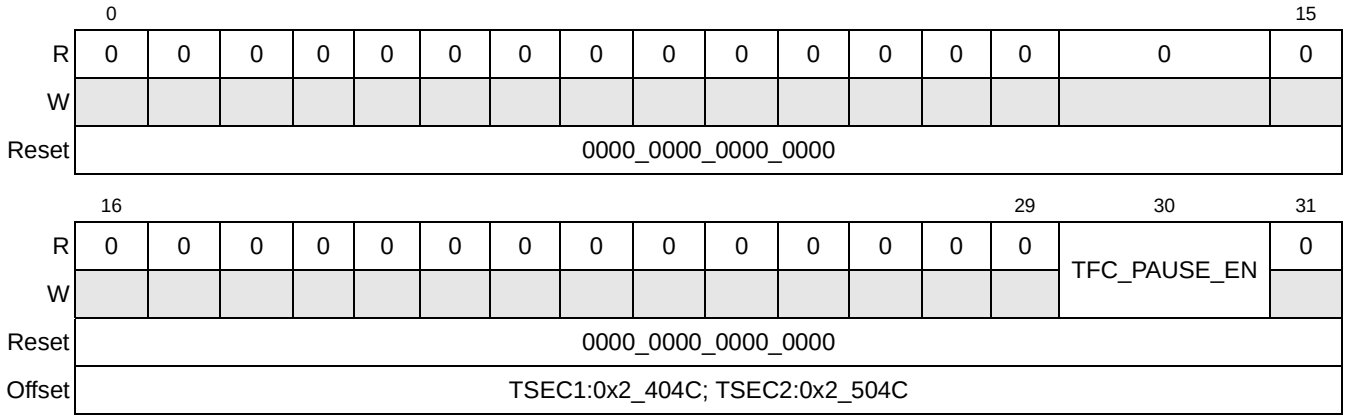


Figure 13-14. FIFO_PAUSE_CTRL Register Definition

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

Table 13-12. FIFO_PAUSE_CTRL Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30	TFC_PAUSE_EN	TFC_PAUSE enable. This bit enables the ability to transmit a pause control frame by setting the TCTRL[TFC_PAUSE] bit. This bit is cleared at reset. 0 Pause control frame transmission disabled. 1 Pause control frame transmission enabled.
31	—	Reserved

13.5.3.2.2 FIFO Transmit Threshold Register (FIFO_TX_THR)

The main purpose of the threshold register is to trigger the unloading of FIFO data to the PHY. It represents the numerical SRAM entry (0-511 for 2-Kbyte FIFO) to trigger the threshold function. If the number of valid entries in the FIFO is equal to or greater than the threshold register, transmission can begin. This register is read/write by software and is initialized to 0000_0000_0000_0000_0000_0001_0000_0000 at system reset. Figure 13-15 shows the FIFO_TX_THR register.

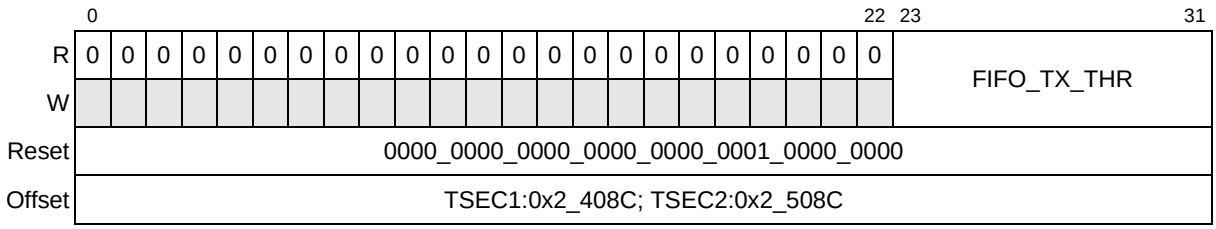


Figure 13-15. FIFO_TX_THR Register Definition

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

Table 13-13. 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.

13.5.3.2.3 FIFO Transmit Starve Register (FIFO_TX_STARVE)

The purpose of the starve register, shown in Figure 13-16, is to inform the system of extremely imminent underrun conditions. It represents the numerical SRAM entry (0-511 for 2-Kbyte FIFO) to trigger the starve function. If the number of valid entries in the FIFO is less than or equal to the starve register, a starve alert is triggered.

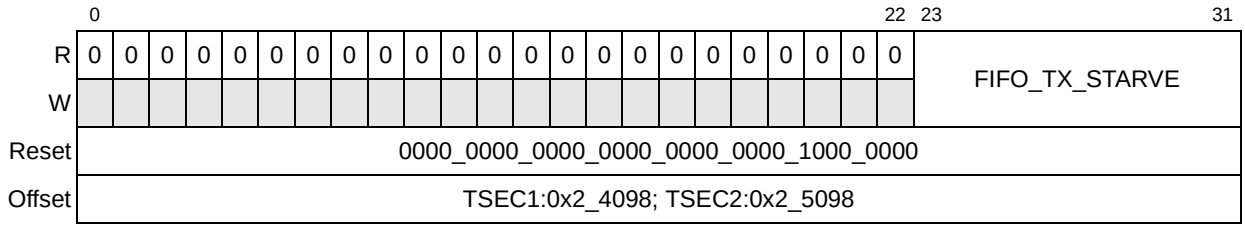


Figure 13-16. FIFO_TX_STARVE Register Definition

Table 13-14 describes the fields of the FIFO_TX_STARVE register.

Table 13-14. 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.

13.5.3.2.4 FIFO Transmit Starve Shutoff Register (FIFO_TX_STARVE_SHUTOFF)

The starve shutoff register, shown in Figure 13-17, contains the watermark level to be used for exiting the starve state. If the starve state is in effect and the number of valid entries in the FIFO becomes greater than or equal to the value in the FIFO transmit starve shutoff register, the starve condition ends. This register is read/write by software.

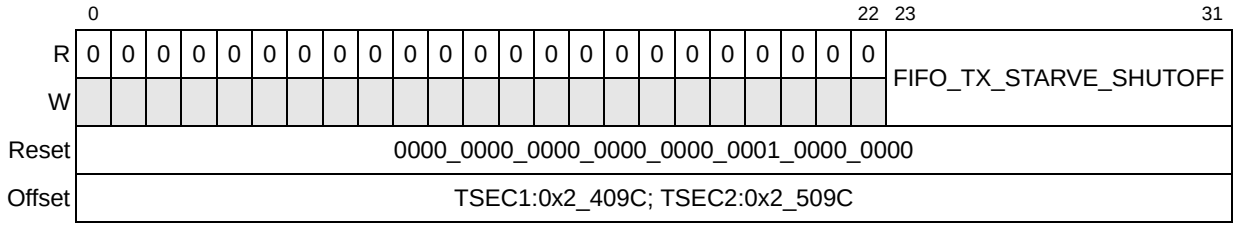


Figure 13-17. FIFO_TX_STARVE_SHUTOFF Register Definition

Table 13-15 describes the fields of the FIFO transmit starve shutoff register.

Table 13-15. FIFO_TX_STARVE_SHUTOFF Field Descriptions

Bits	Name	Description
0–22	—	Reserved
23–31	FIFO_TX_STARVE_SHUTOFF	FIFO transmit starve shutoff. Indicates the value beyond which to exit the starve state. The starve state turns off if the number of valid entries in the FIFO becomes greater than or equal to the FIFO Tx starve shutoff register.

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)

TCTRL, shown in Figure 13-18, is writable by the user to configure the transmit block.

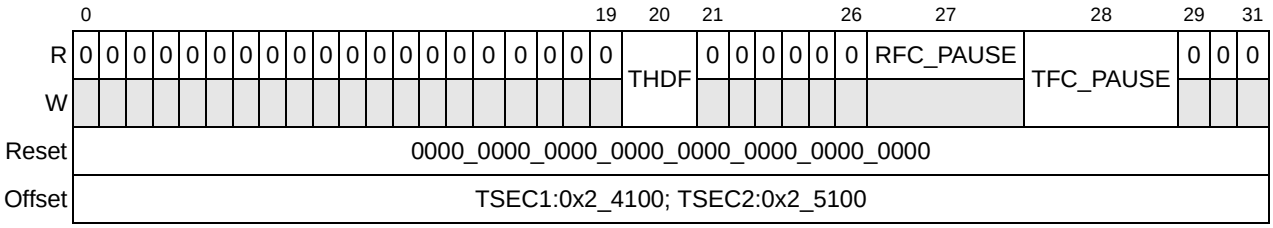


Figure 13-18. TCTRL Register Definition

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

Table 13-16. 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.

Table 13-16. TCTRL Field Descriptions (continued)

Bits	Name	Description
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 Pause duration in progress.
28	TFC_PAUSE	Transmit flow control pause frame. Use this bit to transmit a PAUSE frame. To transmit a flow control pause frame, first set FIFO_PAUSE_CTRL[TFC_PAUSE_EN]. Next, set MACCFG1[GTS]. If TFC_PAUSE is then set, the MAC stops transmission of data frames after the current transmission completes. 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 No outstanding pause frame transmission request. 1 Pause frame transmission requested.
29–31	—	Reserved

13.5.3.3.2 Transmit Status Register (TSTAT)

TSTAT, shown in Figure 13-19, is written by TSEC to convey DMA status information.

	0	1																												31
R	THLT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																															
Reset	0000_0000_0000_0000_0000_0000_0000_0000																														
Offset	TSEC1:0x2_4104; TSEC2:0x2_5104																														

Figure 13-19. TSTAT Register Definition

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

Table 13-17. 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 13-20 shows the TBDLEN register.

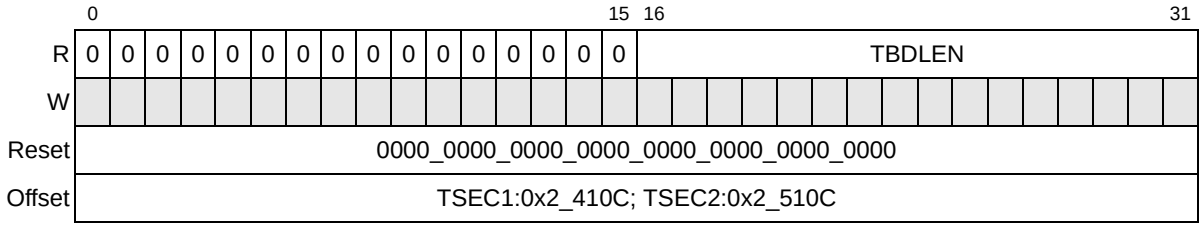


Figure 13-20. TBDLEN Register Definition

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

Table 13-18. TBDLEN Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TBDLEN	Internally written by the DMA module. The transmit channel remains active until TBDLEN is 0.

13.5.3.3.4 Transmit Interrupt Coalescing Configuration Register (TXIC)

TXIC, shown in Figure 13-21, enables and configures the operational parameters for interrupt coalescing associated with transmitted frames. Refer to Section 13.6.2.8.1, “Interrupt Coalescing,” for a functional description of interrupt coalescing.

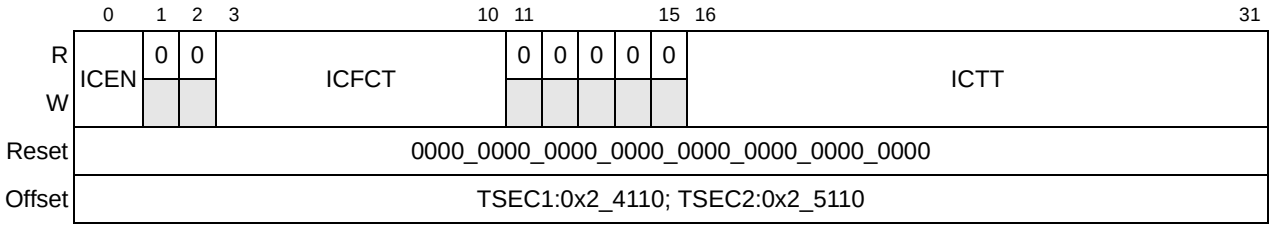


Figure 13-21. TXIC Register Definition

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

Table 13-19. TXIC Field Descriptions

Bits	Name	Description
0	ICEN	Interrupt coalescing enable 0 Interrupt coalescing is disabled. Interrupts are raised as they are received if the TSEC transmit frame interrupt is enabled (IMASK[TXFEN] is set). 1 Interrupt coalescing is enabled. If the TSEC transmit frame interrupt is enabled (IMASK[TXFEN] is set), an interrupt is raised when the threshold number of frames is reached (defined by TXIC[ICFCT]) or when the threshold timer expires (defined by TXIC[ICTT]).
1–2	—	Reserved
3–10	ICFCT	Interrupt coalescing frame count threshold. While interrupt coalescing is enabled (TXIC[ICEN] is set), this value determines how many frames are transmitted before raising an interrupt[1]. Valid values for this field are from 1 to 255. A value of 0 is illegal. If set to 0, an interrupt can only be cleared by first clearing TXIC[ICEN] and then clearing the IEVENT[TXF] bit. Note that a value of 1 functionally defeats the advantages of interrupt coalescing since the frame threshold is reached with each frame transmitted.
11–15	—	Reserved
16–31	ICTT	Interrupt coalescing timer threshold. While interrupt coalescing is enabled (TXIC[ICEN] is set), this value determines the maximum amount of time after transmitting a frame before raising an interrupt[1], subject also to IMASK[TXFEN]. If frames have been transmitted but the frame count threshold has not been met, an interrupt is raised when the threshold timer expires. The threshold timer is reset once an interrupt has been asserted. It begins counting once the interrupt is cleared and IEVENT[TXF] is set. The threshold value is represented in units equal to 64 TSEC interface clocks. Valid values for this field are from 1 to 65535. A value of 0 is illegal. If set to 0, an interrupt can only be cleared by first clearing TXIC[ICEN] and then clearing the IEVENT[TXF] bit.

[1] Interrupts resulting from the Interrupt bit (I) of the buffer descriptor in question and enabled subject to the IMASK register (IMASK[TXFEN] is set).

13.5.3.3.5 Current Transmit Buffer Descriptor Pointer Register (CTBPTR)

CTBPTR contains the address of the transmit buffer descriptor either currently being processed, or processed most recently. Figure 13-22 shows the CTBPTR register.

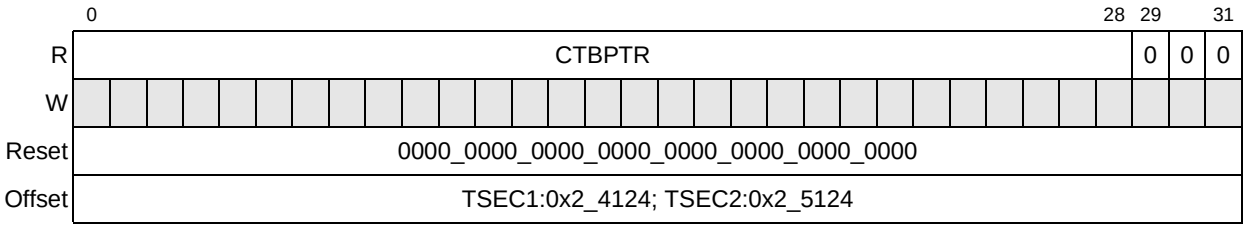


Figure 13-22. CTBPTR Register Definition

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

Table 13-20. CTBPTR Field Descriptions

Bits	Name	Description
0–28	CTBPTR	The CTBPTR register is internally written by the DMA module. The value of this field increments by one (causing the register value to increment by eight) each time a descriptor is read from memory.
29–31	—	Reserved

13.5.3.3.6 Transmit Buffer Descriptor Pointer Register (TBPTR)

TBPTR, shown in Figure 13-23, contains the low-order 32 bits of the next transmit buffer descriptor address. TBPTR takes on the TBASE value when TBASE is written by software. Although not necessary in most applications, the user can modify this register when the transmitter has been gracefully stopped or halted, as indicated by IEVENT[GTSC] or TSTAT[THLT].

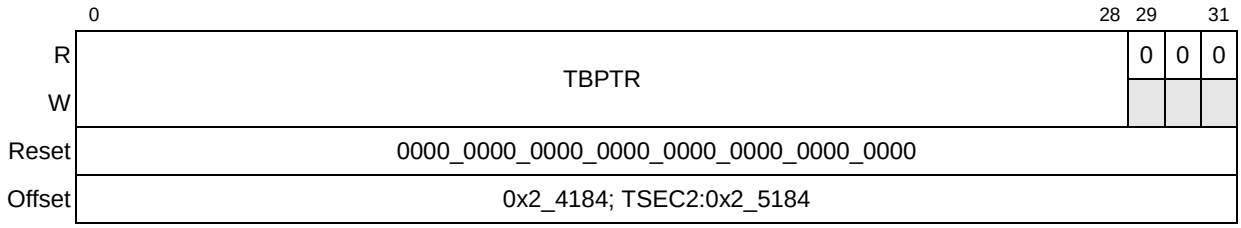


Figure 13-23. TBPTR Register Definition

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

Table 13-21. TBPTR Field Descriptions

Bits	Name	Description
0–28	TBPTR	The TBPTR register is internally written by the DMA module. The value of this field increments by one (causing the register value to increment by eight) each time a descriptor is read from memory.
29–31	—	Reserved

13.5.3.3.7 Transmit Descriptor Base Address Register (TBASE)

The TBASE register, shown in Figure 13-24, is written by the user with the TxBD base address. The value must be divisible by eight for the 8-byte data buffer descriptors.

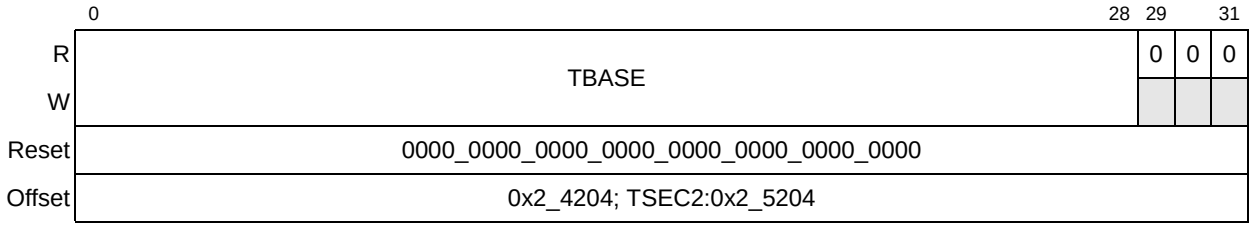


Figure 13-24. TBASE Register Definition

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

Table 13-22. 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

13.5.3.3.8 Out-of-Sequence TxBD Register (OSTBD)

The out-of-sequence TxBD register, OSTBD, shown in Figure 13-25, includes the status/control and data length in the same format as a regular TxBD. It is useful for sending flow control frames. OSTBD[R] is always checked between frames. If it is not ready, a regular frame is sent. If a flow control frame is sent and OSTBD[I] is set, a TXC event is generated after frame transmission. This area must be cleared while not in use. Once the 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.

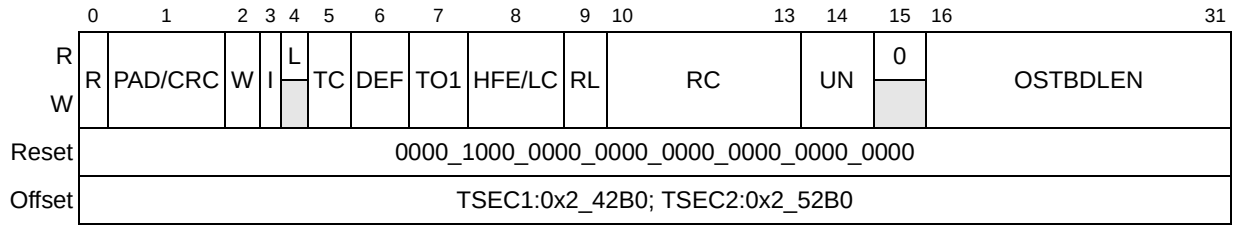


Figure 13-25. OSTBD Register Definition

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

Table 13-23. 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/CRC	Padding and CRC attachment for short frames. (Valid only when MACCFG2[PAD/CRC] is cleared, and MACCFG2[CRC EN] bit is cleared.) If MACCFG2[PAD/CRC] is set, pads are added to all short frames; however, this bit is ignored. 0 Do not add PADs to short frames unless OSTBD[TC] is set. 1 Add PADs to short frames. PAD bytes are inserted until the length of the transmitted frame equals 64 bytes. 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. This bit is ignored by TSEC.
3	I	Interrupt. Written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[TXF] is set after this buffer is serviced. This bit can cause an interrupt if IMASK[TXFEN] is enabled.
4	L	Last in Frame. The OSTBD is always the last in the frame, so L is always set. (Hardwired to a value of 1.)
5	TC	Tx CRC. Written by user. (Valid only while it is set in the first BD and OSTBD[PAD/CRC] is cleared, MACCFG2[PAD/CRC] is cleared, and MACCFG2[CRC EN] is cleared.) If MACCFG2[PAD/CRC] is set or MACCFG2[CRC EN] is set, a CRC is added to all frames and this bit is ignored. 0 End transmission immediately after the last data byte, unless OSTBD[PAD/CRC] is set. 1 Transmit the CRC sequence after the last data byte.
6	DEF	Defer indication. Written by TSEC. Hardware updates this bit after transmitting a frame if used as a ‘Defer indicator.’ Software/user updates this bit while building a transmit buffer descriptor if used as a ‘Hardware Event indicator.’ 0 This frame was not deferred. 1 This frame did not have a collision before it was sent but it was sent late because of deferring.
7	TO1	Transmit software ownership. This read/write bit may be utilized by software, as necessary. Its state does not affect the hardware nor is it affected by the hardware.
8	HFE/LC	Huge frame enable (written by user)/Late collision (written by TSEC) Huge frame enable. Written by user. Valid only while it is set in first BD and the MACCFG2[Huge Frame] is cleared. If MACCFG2[Huge Frame] is set, this bit is ignored. 0 Truncate transmit frame if its length is greater than the MAC’s Maximum Frame Length register. 1 Do not truncate the transmit frame. Late collision. Written by 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.

Table 13-23. OSTBD Field Descriptions (continued)

Bits	Name	Description
10–13	RC	Retry count. Written by TSEC. 0 The frame is sent correctly the first time. 1 More than zero attempts were needed to send the transmit frame. If this field is 15, 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
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 transmits from this BD's data buffer. It is never modified by the TSEC. This field must be greater than zero in order for a transfer to take place.

13.5.3.3.9 Out-of-Sequence Tx Data Buffer Pointer Register (OSTBDP)

The out-of-sequence Tx data buffer pointer register (OSTBDP), shown in Figure 13-26, contains the data buffer pointer fields in the same format as a regular TxBD. With OSTBD, it provides the complete 8-byte descriptor. This area must be cleared while not in use.

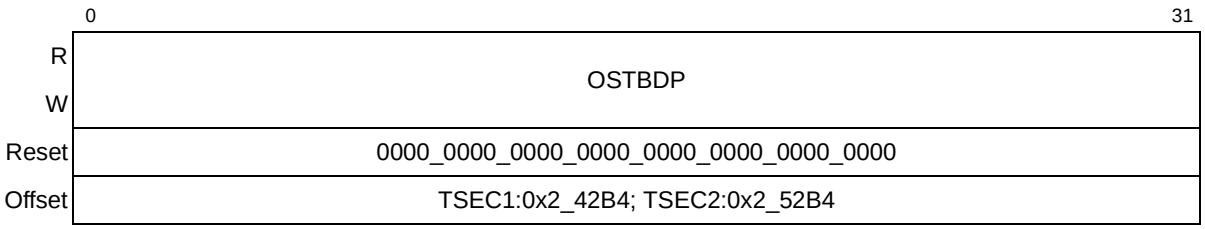


Figure 13-26. OSTBDP Register Definition

Table 13-24 describes OSTBDP.

Table 13-24. 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)

RCTRL, shown in Figure 13-27, 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.

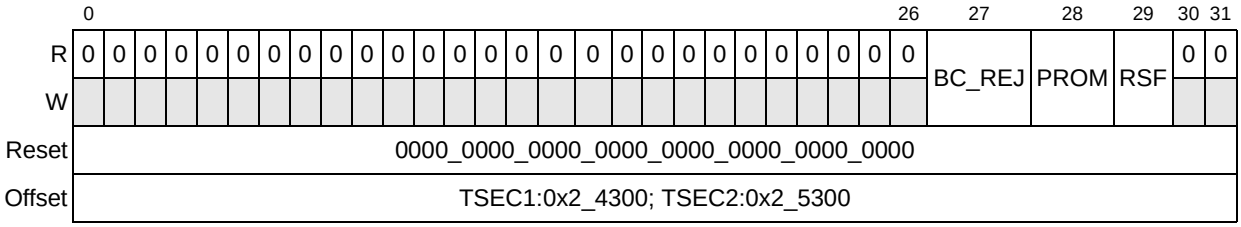


Figure 13-27. RCTRL Register Definition

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

Table 13-25. RCTRL Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	BC_REJ	Broadcast frame reject. If this bit is set, frames with DA (destination address) = FFFF_FFFF_FFFF are rejected unless RCTRL[PROM] is set. If both BC_REJ and RCTRL[PROM] are set, then frames with broadcast DA are accepted and the M (MISS) bit is set in the receive BD. 0 Broadcast frame reject is disabled. 1 Broadcast frame reject is enabled.
28	PROM	Promiscuous mode. When set, all frames except pause frames are accepted. 0 Promiscuous mode is disabled. 1 Promiscuous mode is enabled.
29	RSF	Receive short frame mode. When set, enables the reception of frames shorter than MINFLR bytes. Note that in order for short frames to be received when RSF is set, a DA hit must occur. When RSF is cleared, all frames shorter than MINFLR are automatically rejected. 0 Receive short fram mode is disabled. 1 Receive short frame mode is enabled.
30–31	—	Reserved

13.5.3.4.2 Receive Status Register (RSTAT)

TSEC writes to RSTAT, shown in Figure 13-28, 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 must clear the QHLT bit to take the TSEC receiver function out of a halt state.

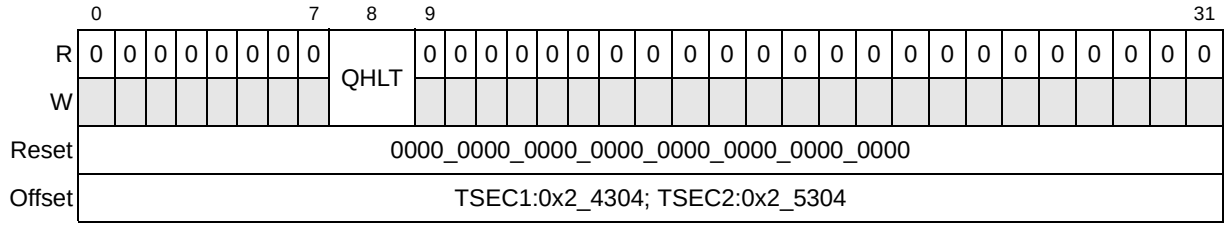


Figure 13-28. RSTAT Register Definition

Table 13-26 describes the fields of the RSTAT register.

Table 13-26. RSTAT Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8	QHLT	RxBD queue is halted. When IEVENT[BSY] or IEVENT[EBERR] is set during reception of a packet, RSTAT[QHLT] is also set. In order to begin receiving packets again, the user must clear RSTAT[QHLT]. This bit is set whenever the TSEC reads an RxBD with its Empty field cleared or encounters a system bus error while processing an RX packet. It is a hardware-initiated stop indication (DMA_CTRL[GRS] being set by the user does not cause this bit to be set.). The current frame and all other frames directed to the halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 RxBD queue is enabled for Ethernet reception. (That is, it is not halted.) 1 All Ethernet controller receive activity to RxBD queue is halted.
9–31	—	Reserved

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 13-29 shows the RBDLEN register.

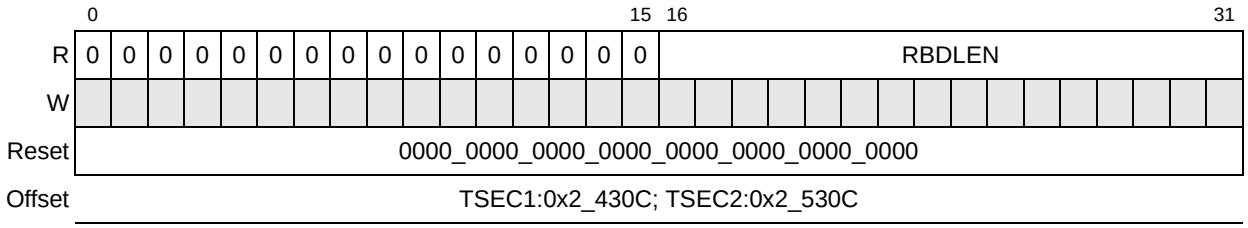


Figure 13-29. RBDLEN Register Definition

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

Table 13-27. 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.

13.5.3.4.4 Receive Interrupt Coalescing Configuration Register (RXIC)

The RXIC register enables and configures the operational parameters for interrupt coalescing associated with received frames. Refer to Section 13.6.2.8.1, “Interrupt Coalescing,” for a functional description of interrupt coalescing as additional details regarding the use of this register. Figure 13-30 shows the RXIC register.

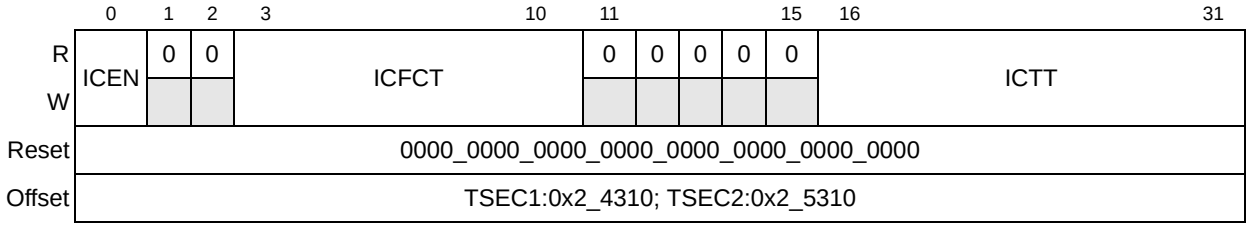


Figure 13-30. RXIC Register Definition

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

Table 13-28. RXIC Field Descriptions

Bits	Name	Description
0	ICEN	Interrupt coalescing enable 0 Interrupt coalescing is disabled. Interrupts are raised as they are received if the TSEC receive frame interrupt is enabled (IMASK[RXFEN] is set). 1 Interrupt coalescing is enabled. If the TSEC receive frame interrupt is enabled (IMASK[RXFEN] is set), an interrupt is raised when the threshold number of frames is reached (defined by RXIC[ICFCT]) or when the threshold timer expires (defined by RXIC[ICTT]).
1–2	—	Reserved
3–10	ICFCT	Interrupt coalescing frame count threshold. While interrupt coalescing is enabled (RXIC[ICEN] is set), this value determines how many frames are received before raising an interrupt[1]. Valid values for this field are from 1 to 255. A value of 0 is illegal. If set to 0, an interrupt can only be cleared by first clearing RXIC[ICEN] and then clearing the IEVENT[RXF] bit. Note that a value of 1 functionally defeats the advantages of interrupt coalescing since the frame threshold is reached with each frame received.
11–15	—	Reserved
16–31	ICTT	Interrupt coalescing timer threshold. While interrupt coalescing is enabled (RXIC[ICEN] is set), this value determines the maximum amount of time after receiving a frame before raising an interrupt[1], subject also to IMASK[RXFEN]. If frames have been received but the frame count threshold has not been met, an interrupt is raised when the threshold timer expires. The threshold timer is reset once an interrupt has been asserted. It begins counting once the interrupt is cleared and IEVENT[RXF] is set. The threshold value is represented in units equal to 64 TSEC interface clocks. Valid values for this field are from 1 to 65535. A value of 0 is illegal. If set to 0, an interrupt can only be cleared by first clearing RXIC[ICEN] and then clearing the IEVENT[RXF] bit.

[1] Interrupts resulting from the Interrupt bit (I) of the buffer descriptor in question and enabled subject to the IMASK register (IMASK[RXFEN] is set).

13.5.3.4.5 Current Receive Buffer Descriptor Pointer Register (CRBPTR)

CRBPTR contains the address of the receive buffer descriptor either currently being processed, or processed most recently. Figure 13-31 shows the CRBPTR register.

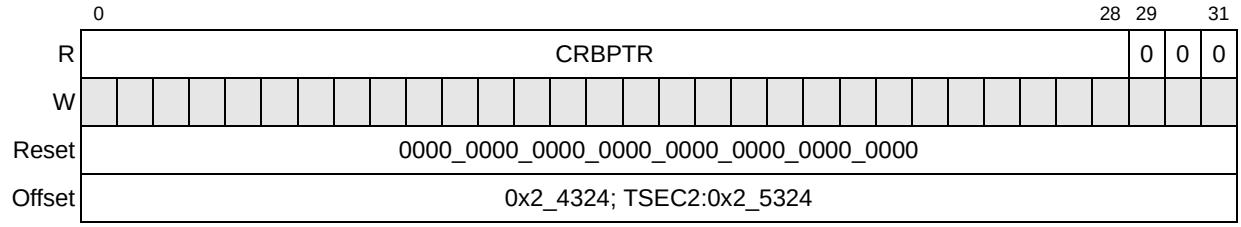


Figure 13-31. CRBPTR Register Definition

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

Table 13-29. CRBPTR Field Descriptions

Bits	Name	Description
0–28	CRBPTR	The CRBPTR register is internally written by the DMA module. The value of this field increments by one (causing the register value to increment by eight) each time a descriptor is read from memory.
29–31	—	Reserved

13.5.3.4.6 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 13-32 shows the MRBLR.

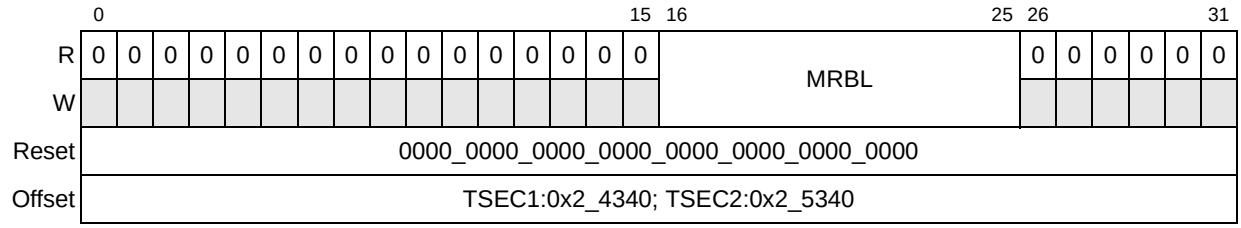


Figure 13-32. MRBL Register Definition

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

Table 13-30. 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 must be cleared.

13.5.3.4.7 Receive Buffer Descriptor Pointer Register (RBPTR)

RBPTR contains the receive buffer descriptor address. Figure 13-33 shows the RBPTR register. This register takes on the value of RBASE when the RBASE register is written by software. Although not necessary in most applications, the user can modify this register when the transmitter has been gracefully stopped or halted, as indicated by TSTAT[THLT].

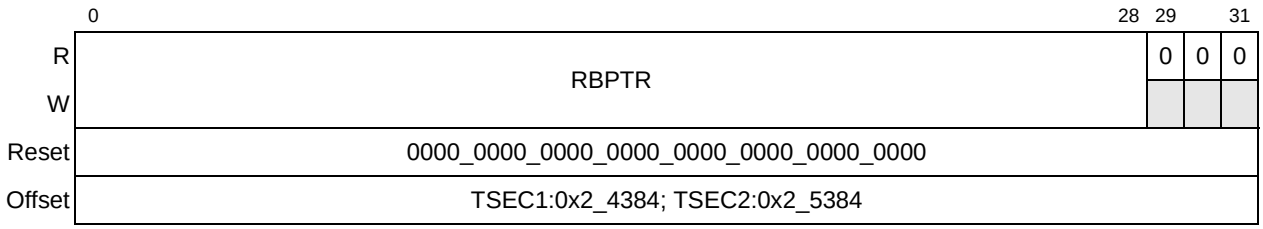


Figure 13-33. RBPTR Register Definition

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

Table 13-31. RBPTR Field Descriptions

Bits	Name	Description
0–28	RBPTR	The RBPTR register is internally written by the DMA module. The value of this field increments by one (causing the register value to increment by eight) each time a descriptor is read from memory.
29–31	—	Reserved

13.5.3.4.8 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 13-34 shows the RBASE register.

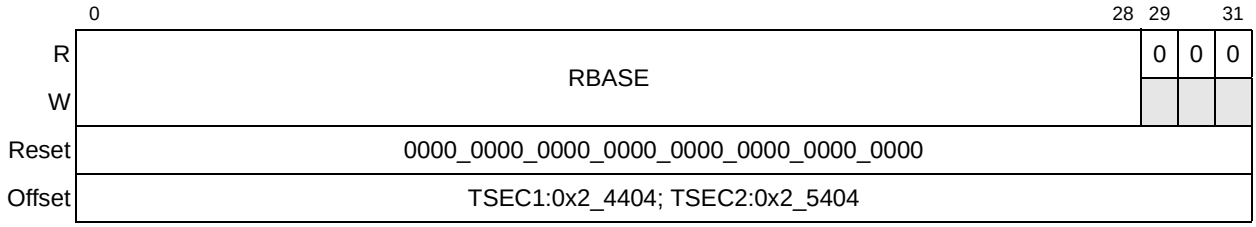


Figure 13-34. RBASE Register Definition

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

Table 13-32. RBASE Field Descriptions

Bits	Name	Description
0–28	RBASE	Receive base. RBASE defines the starting location in the memory map for the TSEC RxBD.
29–31	—	Reserved. These bits must be set to zero, to cause the value of RBASE to be a multiple of eight.

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

MAC configuration registers 1 and 2 provide a way to configure the MAC in multiple ways:

- Adjusting the preamble length—The length of the preamble can be adjusted from the nominal seven bytes to some other (non-zero) value.
- Varying pad/CRC combinations—Three pad/CRC combinations are provided to handle a variety of system requirements. The most simple are frames that already have a valid FCS field. In this case, the CRC is checked and reported 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 must 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:

$$0 \leq r \leq 2^k, \text{ where } k = \min(n,10).$$

So, after the first collision, TSEC backs-off either 0 or 1 slot times. After the fifth collision, TSEC backs-off between 0 and 32 slot times. After the tenth collision, the maximum number of slot times to back-off is 1024. This can be adjusted through 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 1-Gbps 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. 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 indicator registers) are used to exercise this interface between a host processor and one or more PHY devices (including the TBI). External PHYs may only be configured using the MII management interface of TSEC1 since the interface signals (EC_MDC and EC_MDIO) are only driven by TSEC1.

The TSEC MII’s registers provide the ability to perform continuous read cycles, called a scan cycle. If requested (by setting MIIMCOM[Scan Cycle]), the part performs repetitive read cycles of the PHY status register, for example. In this way, link characteristics may be monitored more efficiently. The different fields in the MII management indicator register (scan, not valid and busy) are used to indicate availability of each read of the scan cycle to the host from MIIMSTAT[PHY Status].

Yet another parameter that can be modified through the MII registers is the length of the MII management interface preamble. After establishing that a PHY supports preamble suppression, the host may so configure the 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

The following are the MAC registers.

13.5.3.6.1 MAC Configuration Register 1 (MACCFG1)

The MACCFG1 register is written by the user. Figure 13-35 shows the MACCFG1 register.

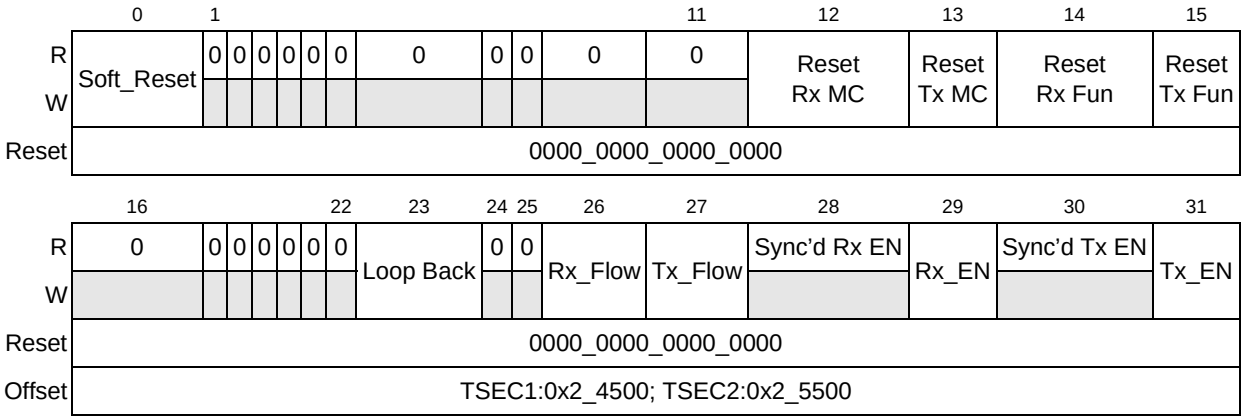


Figure 13-35. MACCFG1 Register Definition

Table 13-33 describes the fields of the MACCFG1 register.

Table 13-33. 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," for more information on setting this bit. 0 Normal operation 1 Place all modules within the MAC in reset.
1–11	—	Reserved
12	Reset Rx MC	Reset receive MAC control block. This bit is cleared by default. 0 Normal operation 1 Place the receive MAC control block in reset. This block detects control frames and contains the pause timers.
13	Reset Tx MC	Reset transmit MAC control block. This bit is cleared by default. 0 Normal operation 1 Place the PETMC transmit MAC control block in reset. This block multiplexes data and control frame transfers. It also responds to XOFF PAUSE control frames.

Table 13-33. MACCFG1 Field Descriptions (continued)

Bits	Name	Description
14	Reset Rx Fun	Reset receive function block. This bit is cleared by default. 0 Normal operation 1 Place the receive function block in reset. This block performs the receive frame protocol.
15	Reset Tx Fun	Reset transmit function block. This bit is cleared by default. 0 Normal operation 1 Place the transmit function block in reset. This block performs the frame transmission protocol.
16–22	—	Reserved
23	Loop Back	Loop back. This bit is cleared by default. 0 Normal operation 1 Loop back the MAC transmit outputs to the MAC receive inputs.
24–25	—	Reserved
26	Rx_Flow	Receive flow. This bit is cleared by default. 0 The receive MAC control ignores PAUSE flow control frames. 1 The receive MAC control detects and acts on PAUSE flow control frames.
27	Tx_Flow	Transmit flow. This bit is cleared by default. 0 The transmit MAC control may not send PAUSE flow control frames if requested by the system. 1 The transmit MAC control may send PAUSE flow control frames if requested by the system.
28	Sync'd Rx EN	Receive enable synchronized to the receive stream (Read-only) 0 Frame reception is not enabled. 1 Frame reception is enabled.
29	Rx_EN	Receive enable. This bit is cleared by default. If set, prior to clearing this bit, set DMACTRL[GRS] then confirm subsequent occurrence of the graceful receive stop interrupt (IEVENT[GRSC] is set). 0 The MAC may not receive frames from the PHY. 1 The MAC may receive frames from the PHY.
30	Sync'd Tx EN	Transmit enable synchronized to the transmit stream (Read-only) 0 Frame transmission is not enabled. 1 Frame transmission is enabled.
31	Tx_EN	Transmit enable. This bit is cleared by default. If set, prior to clearing this bit, set DMACTRL[GTS] then confirm subsequent occurrence of the graceful transmit stop interrupt (IEVENT[GTSC] is set). 0 The MAC may not transmit frames from the system. 1 The MAC may transmit frames from the system.

13.5.3.6.2 MAC Configuration Register 2 (MACCFG2)

The MACCFG2 register is written by the user. Figure 13-36 shows the MACCFG2 register.

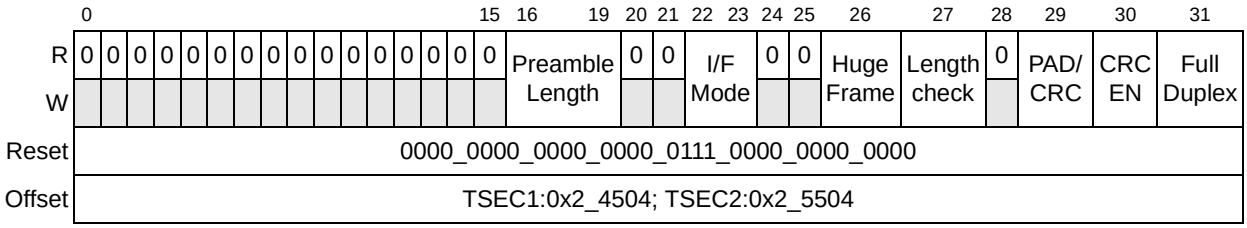


Figure 13-36. MACCFG2 Register Definition

Table 13-34 describes the fields of the MACCFG2 register.

Table 13-34. MACCFG2 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–19	Preamble Length	Determines the length in bytes of the preamble field of the packet. Its default is 0x7. A preamble length of 0 is not supported.
20–21	—	Reserved
22–23	I/F Mode	Determines the type of interface to which the MAC is connected. Its default is 00. 00 Reserved 01 Nibble mode (MII) 10 Byte mode (GMII/TBI) 11 Reserved
24–25	—	Reserved
26	Huge Frame	Huge frame enable. Cleared by default. 0 Limit the length of frames to the MAXIMUM FRAME LENGTH value. 1 Frames longer than the MAXIMUM FRAME LENGTH may be transmitted and received.
27	Length check	Length check. This bit is cleared by default. 0 No length field checking is performed. 1 The MAC checks the frame's length field to ensure it matches the actual data field length.
28	—	Reserved
29	PAD/CRC	Pad and append CRC. This bit is cleared by default. 0 Frames presented to the MAC have a valid length and contain a CRC. 1 The MAC pads all transmitted short frames and appends a CRC to every frame regardless of padding requirement.
30	CRC EN	CRC enable. If the configuration bit PAD/CRC or the per-packet PAD/CRC bit is set, CRC EN is ignored. This bit is cleared by default. 0 Frames presented to the MAC have a valid length and contain a valid CRC. 1 The MAC appends a CRC on all frames. Clear this bit if frames presented to the MAC have a valid length and contain a valid CRC.
31	Full Duplex	Full duplex configure. This bit is cleared by default. 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 13-37 shows the IPGIFG register.

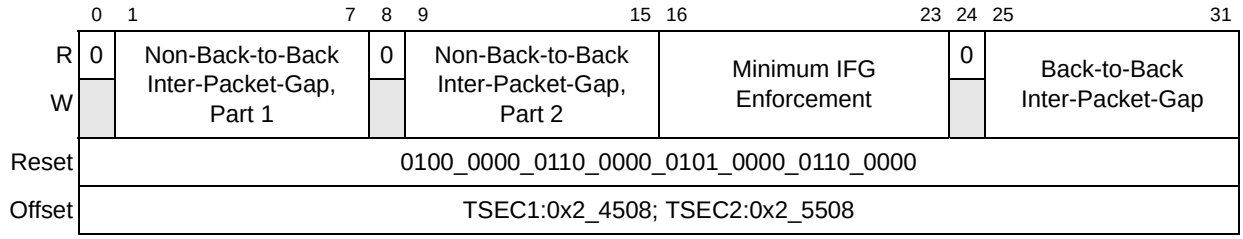


Figure 13-37. IPGIFG Register Definition

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

Table 13-35. IPGIFG Field Descriptions

Bits	Name	Description
0	—	Reserved
1–7	Non-Back-to-Back Inter-Packet-Gap, Part 1	Programmable field representing the optional carrier Sense window referenced in IEEE 802.3/4.2.3.2.1 ‘carrier deference’. If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision; thus, ensuring fair access to medium. Its range of values is 0x00 to IPGR2. Its default is 0x40 (64d) which follows the two-thirds/one-third guideline.
8	—	Reserved
9–15	Non-Back-to-Back Inter-Packet-Gap, Part 2	Programmable field representing the non-back-to-back inter-packet-gap in bits. Its default is 0x60 (96d), which represents the minimum IPG of 96 bits.
16–23	Minimum IFG Enforcement	Programmable field representing the minimum number of bits of IFG to enforce between frames. A frame is dropped whose IFG is less than that programmed. The default setting of 0x50 (80d) represents half of the nominal minimum IFG which is 160 bits.
24	—	Reserved
25–31	Back-to-Back Inter-Packet-Gap	Programmable field representing the IPG between back-to-back packets. This is the IPG parameter used exclusively in full-duplex mode and in half-duplex mode if two transmit packets are sent back-to-back. Set this field to the number of bits of IPG desired. The default setting of 0x60 (96d) represents the minimum IPG of 96 bits.

13.5.3.6.4 Half-Duplex Register (HAFDUP)

The HAFDUP register is written by the user. Figure 13-38 shows the HAFDUP register.

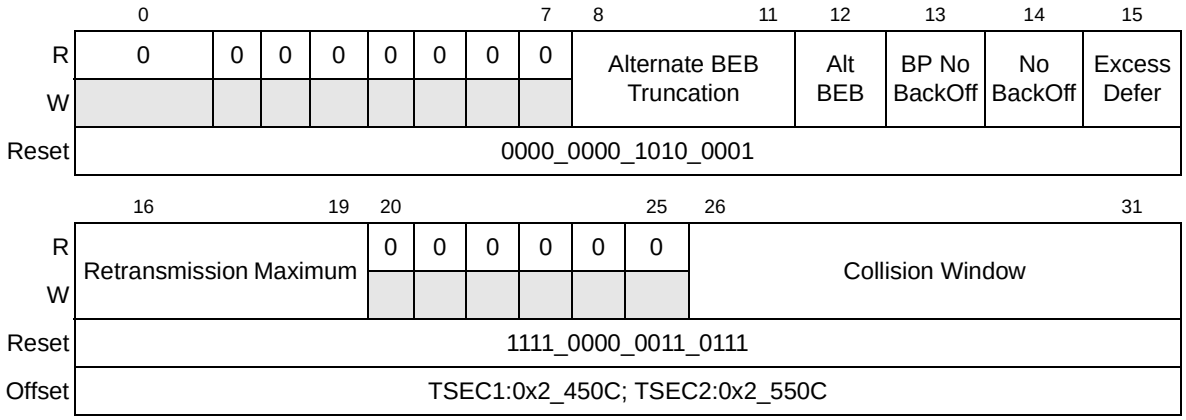


Figure 13-38. Half-Duplex Register Definition

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

Table 13-36. HAFDUP Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	Alternate BEB Truncation	This field is used while alternate binary exponential back-off enable is set. The value programmed is substituted for the Ethernet standard value of ten. Its default is 0xA.
12	Alt BEB	Alternate binary exponential back-off. This bit is cleared by default. 0 The Tx MAC follows the standard binary exponential back-off rule. 1 The Tx MAC uses the alternate binary exponential back-off truncation setting instead of the 802.3 standard tenth collision. The standard specifies that any collision after the tenth uses one less than 2^{10} ($2^{10} – 1$) as the maximum back-off time.
13	BP No BackOff	Back pressure no back-off. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back-off rule. 1 The Tx MAC immediately re-transmits, following a collision, during back pressure operation.
14	No BackOff	No back-off. This bit is cleared by default. 0 The Tx MAC follows the binary exponential back-off rule. 1 The Tx MAC immediately re-transmits following a collision.
15	Excess Defer	Excessively deferred. This bit is set by default. 0 The Tx MAC aborts the transmission of a packet that is excessively deferred. 1 The Tx MAC allows the transmission of a packet that is excessively deferred.
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.

Table 13-36. HAFDUP Field Descriptions (continued)

Bits	Name	Description
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 13-39 shows the MAXFRM register.

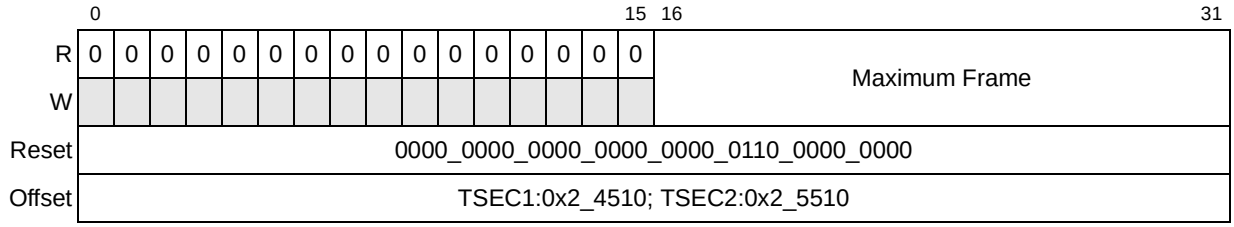


Figure 13-39. Maximum Frame Length Register Definition

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

Table 13-37. MAXFRM Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	Maximum Frame	By default this field is set to 0x0600 (1536d). It sets the maximum frame size in both the transmit and receive directions. (Refer to MACCFG2[Huge Frame].)

13.5.3.6.6 MII Management Configuration Register (MIIMCFG)

The MIIMCFG register is written by the user. Figure 13-40 shows the MIIMCFG register.

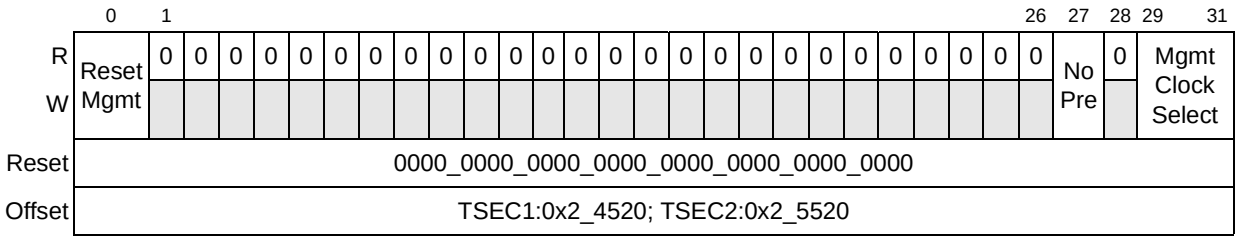


Figure 13-40. MII Management Configuration Register Definition

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

Table 13-38. MIIMCFG Field Descriptions

Bits	Name	Description
0	Reset Mgmt	Reset management. This bit is cleared by default. 0 Allow the MII Mgmt to perform management read/write cycles if requested. 1 Reset the MII Mgmt.
1–26	—	Reserved
27	No Pre	Preamble suppress. This bit is cleared by default. 0 The MII Mgmt performs management read/write cycles with 32 clocks of preamble. 1 The MII Mgmt suppresses preamble generation and reduces the management cycle from 64 clocks to 32 clocks. This is in accordance with IEEE 802.3/22.2.4.4.2.
28	—	Reserved
29–31	Mgmt Clock Select	This field determines the clock frequency of the Mgmt Clock (EC_MDC). Its default value is 000. The source clock frequency is equal to the core complex bus clock divided first by eight and then futher divided by the following value: 000 Source clock divided by 4 001 Source clock divided by 4 010 Source clock divided by 6 011 Source clock divided by 8 100 Source clock divided by 10 101 Source clock divided by 14 110 Source clock divided by 20 111 Source clock divided by 28

13.5.3.6.7 MII Management Command Register (MIIMCOM)

The MIIMCOM register is written by the user. Figure 13-41 shows the MIIMCOM register.

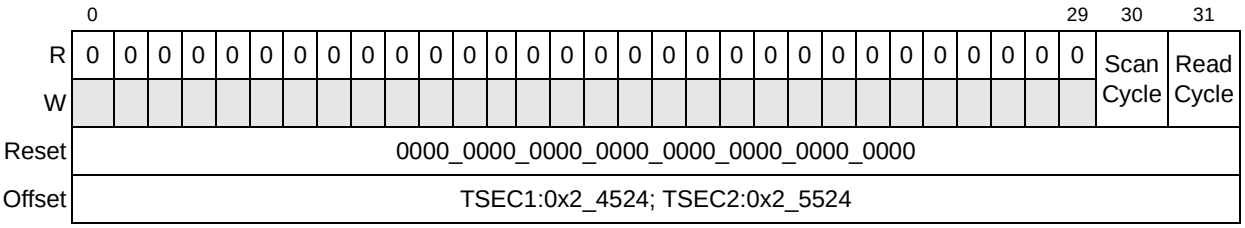


Figure 13-41. MIIMCOM Register Definition

Table 13-39 describes the fields of the MIIMCOM register.

Table 13-39. MIIMCOM Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30	Scan Cycle	Scan cycle. This bit is cleared by default. 0 Normal operation. 1 The MII management continuously performs read cycles. This is useful for polling a PHY register, for example, monitoring link fails. Data is returned in register MIIMSTAT[PHY Status].
31	Read Cycle	Read cycle. This bit is cleared by default but is not self-clearing once set. 0 Normal operation. 1 The MII management performs a single read cycle upon the transition of this bit from 0 to 1 using the PHY address (at MIIMADD[PHY Address]) and the register address (at MIIMADD[Register Address]). The 0 to 1 transition of this bit also causes the MIIMIND[BUSY] bit to be set. The read is complete when the MIIMIND[BUSY] bit clears. Data is returned in register MIIMSTAT[PHY Status].

13.5.3.6.8 MII Management Address Register (MIIMADD)

The MIIMADD register is written by the user. Figure 13-42 shows the MIIMADD register.

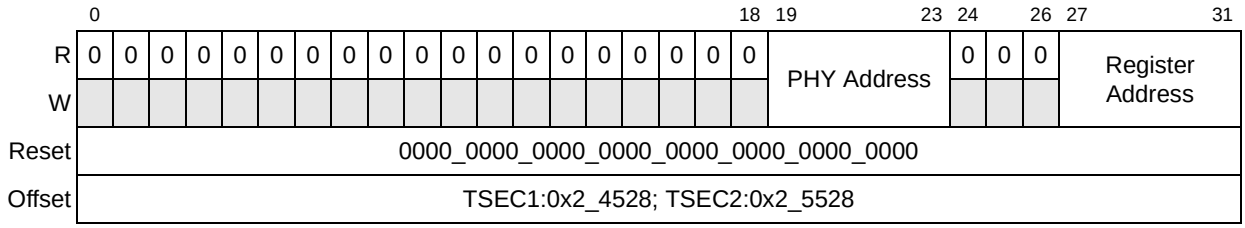


Figure 13-42. MIIMADD Register Definition

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

Table 13-40. MIIMADD Field Descriptions

Bits	Name	Description
0–18	—	Reserved
19–23	PHY Address	This field represents the 5-bit PHY address field of management cycles. Up to 31 PHYs can be addressed (0 is reserved).
24–26	—	Reserved
27–31	Register Address	This field represents the 5-bit register address field of management cycles. Up to 32 registers can be accessed. Its default value is 0x00.

13.5.3.6.9 MII Management Control Register (MIIMCON)

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

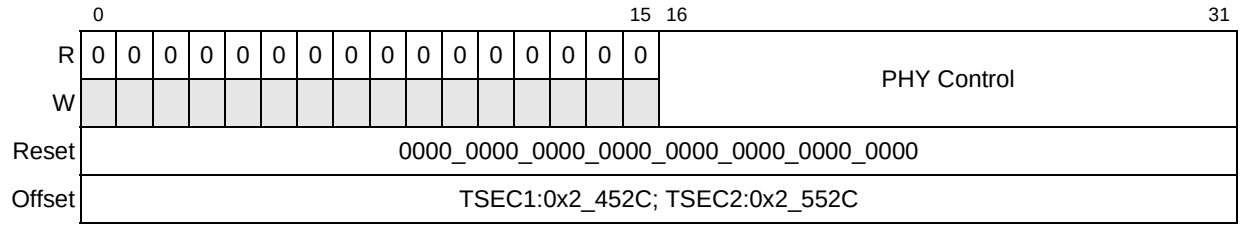


Figure 13-43. MII Management Control Register Definition

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

Table 13-41. 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, shown in Figure 13-44, is read-only by the user.

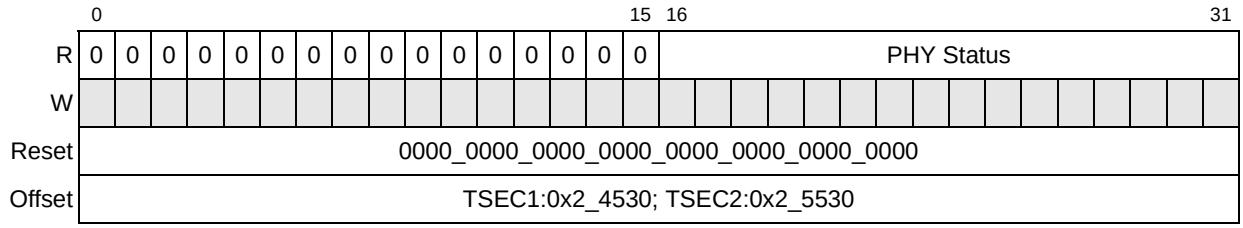


Figure 13-44. MIIMSTAT Register Definition

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

Table 13-42. 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 13-45 shows the MIIMIND register.

	0	28	29	30	31
R	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0	Not Valid	Scan	Busy
W					
Reset	0000_0000_0000_0000_0000_0000_0000_0000				
Offset	TSEC1:0x2_4534; TSEC2:0x2_5534				

Figure 13-45. MII Management Indicator Register Definition

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

Table 13-43. 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)

Figure 13-46 shows the IFSTAT register.

	0	21	22	23	27	28	29	31
R	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0	Excess Defer	0 0 0 0 0	0	Link Fail	0	0 0
W								
Reset	0000_0000_0000_0000_0000_0000_0000_0000							
Offset	TSEC1:0x2_453C; TSEC2:0x2_553C							

Figure 13-46. Interface Status Register Definition

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

Table 13-44. 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.
23–27	—	Reserved
28	Link Fail	Link Fail. This bit indicates the status of signal detection. 0 The100X module has detected a “signal detect” for longer than 330 mS. 1 The100X module has detected a “signal detect” for less than 330 mS or not at all.
29–31	—	Reserved

13.5.3.6.13 Station Address Register Part 1 (MACSTNADDR1)

The MACSTNADDR1 register is written by the user. Figure 13-47 shows the MACSTNADDR1 register. The value of the station address written by the user into MACSTNADDR1 and MACSTNADDR2 is byte-reversed from how it would appear in the DA field of a frame in memory. For example, for a station address of 0x12345678ABCD, perform a write to MACSTNADDR1 of 0xCDAB7856, and to MACSTNADDR2 of 0x34120000. When the user reads MACSTNADDR1, 0xCDAB7856 is returned. A read of MACSTNADDR2 returns a value of 0x34120000. Note, the I/G and U/L bits of the frame’s DA field is located at the LSBs of the 1st octet stored in MACSTNADDR2, where the I/G bit is bit 15, and the U/L bit is bit 14.

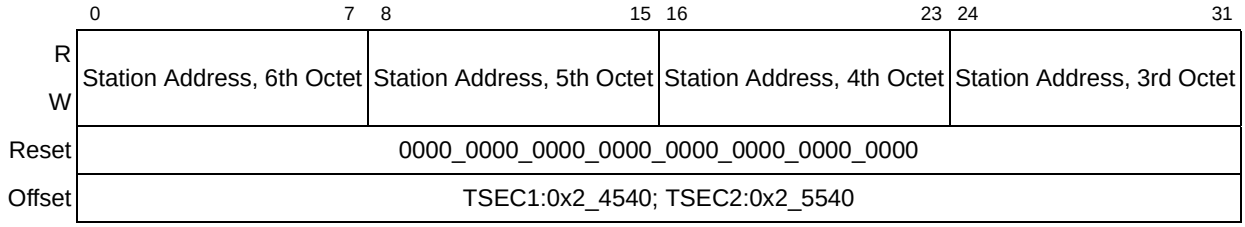


Figure 13-47. Station Address Part 1 Register Definition

Table 13-45 describes the fields of the MACSTNADDR1 register.

Table 13-45. MACSTNADDR1 Field Descriptions

Bits	Name	Description
0–7	Station Address, 6th Octet	This field holds the sixth octet of the station address. The sixth octet (station address bits 40–47) defaults to a value of 0x00.
8–15	Station Address, 5th Octet	This field holds the fifth octet of the station address. The fifth octet (station address bits 32–39) defaults to a value of 0x00.
16–23	Station Address, 4th Octet	This field holds the fourth octet of the station address. The fourth octet (station address bits 24–31) defaults to a value of 0x00.
24–31	Station Address, 3rd Octet	This field holds the third octet of the station address. The third octet (station address bits 16–23) defaults to a value of 0x00.

13.5.3.6.14 Station Address Register Part 2 (MACSTNADDR2)

The MACSTNADDR2 register is written by the user. Figure 13-48 shows the MACSTNADDR2 register. Note, the I/G and U/L bits of the frame's DA field is located at the LSBs of the 1st octet stored in MACSTNADDR2, where the I/G bit is bit 15, and the U/L bit is bit 14.

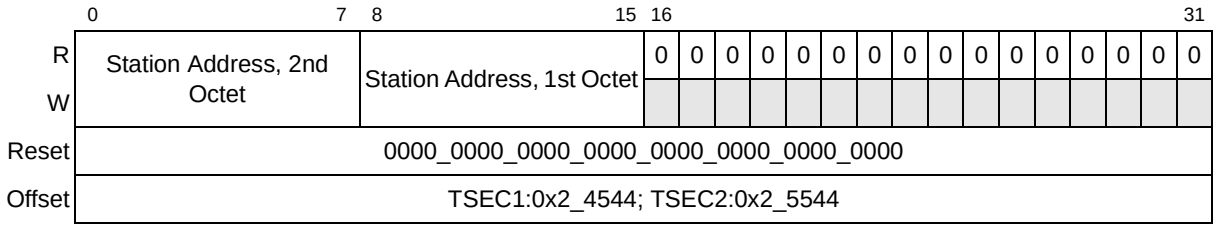


Figure 13-48. Station Address Part 2 Register Definition

Table 13-46 describes the fields of the MACSTNADDR2 register.

Table 13-46. MACSTNADDR2 Field Descriptions

Bits	Name	Description
0–7	Station Address, 2nd Octet	This field holds the second octet of the station address. The second octet (station address bits 8–15) defaults to a value of 0x00.
8–15	Station Address, 1st Octet	This field holds the first octet of the station address. The first octet (station address bits 0–7) defaults to a value of 0x00.
16–31	—	Reserved

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 are 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 signal. Figure 13-49 shows the TR64 register.

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

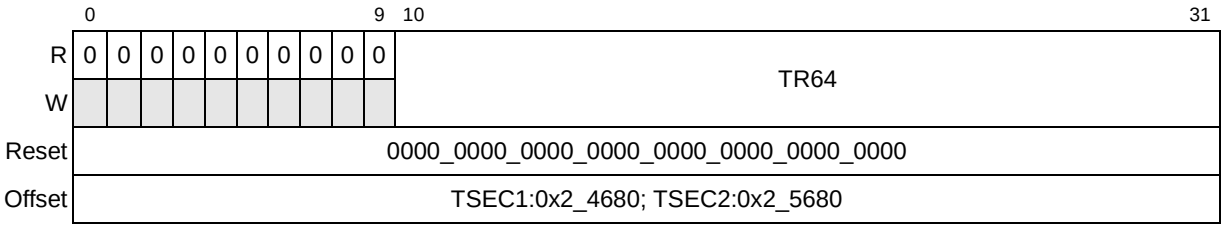


Figure 13-49. Transmit and Receive 64-Byte Frame Register Definition

Table 13-47 describes the fields of the TR64 register.

Table 13-47. 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 Register (TR127)

Figure 13-50 shows the TR127 register.

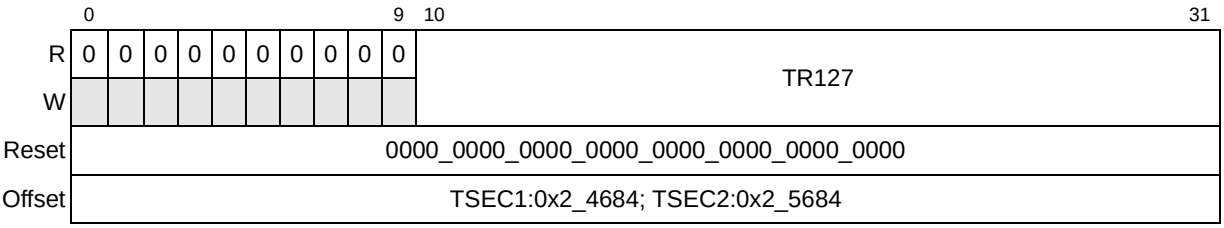


Figure 13-50. Transmit and Receive 65- to 127-Byte Frame Register Definition

Table 13-48 describes the fields of the TR127 register.

Table 13-48. 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 Register (TR255)

Figure 13-51 shows the TR255 register.

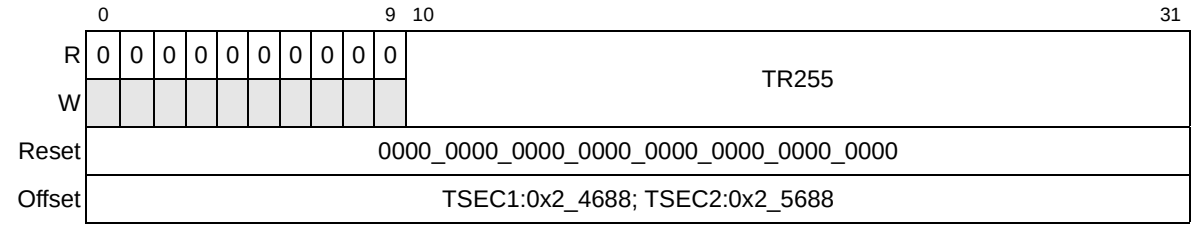


Figure 13-51. Transmit and Receive 128- to 255-Byte Frame Register Definition

Table 13-49 describes the fields of the TR255 register.

Table 13-49. 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 Register (TR511)

Figure 13-52 shows the TR511 register.

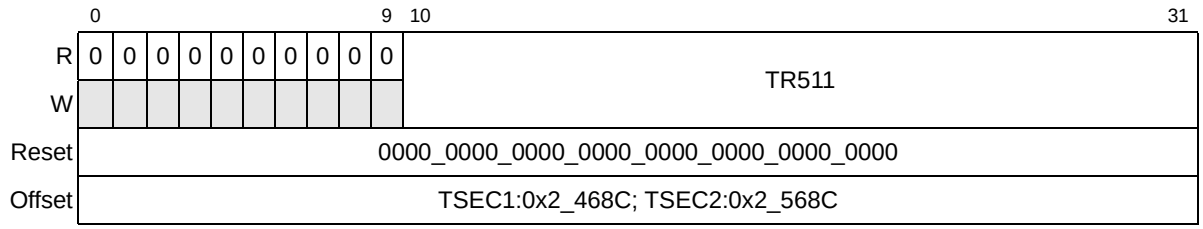


Figure 13-52. Transmit and Receive 256- to 511-Byte Frame Register Definition

Table 13-50 describes the fields of the TR511 register.

Table 13-50. 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 Register (TR1K)

Figure 13-53 shows the TR1K register.

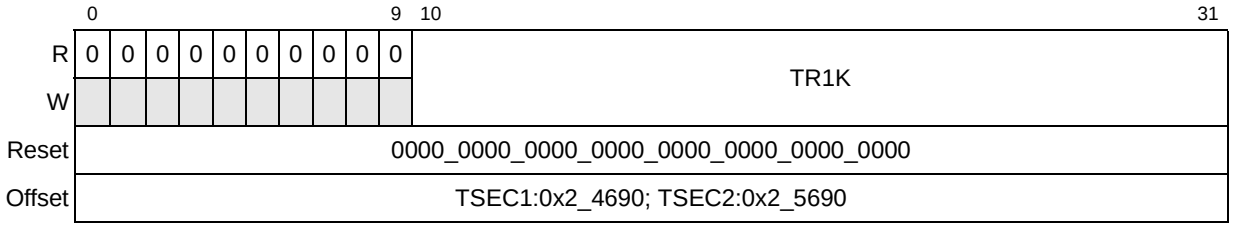


Figure 13-53. Transmit and Receive 512- to 1023-Byte Frame Register Definition

Table 13-51 describes the fields of the TR1K register.

Table 13-51. 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 Register (TRMAX)

Figure 13-54 shows the TRMAX register.

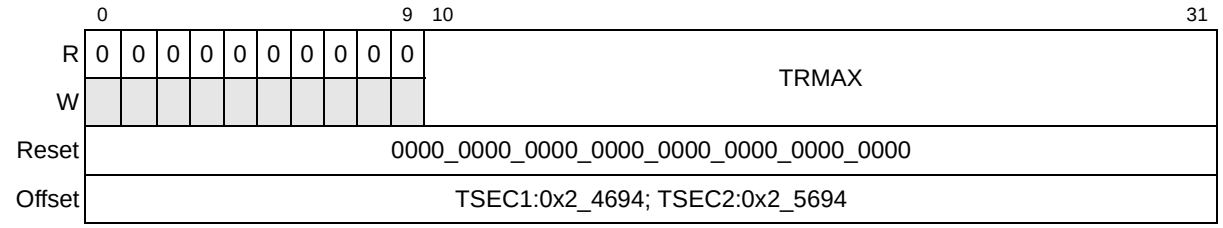


Figure 13-54. Transmit and Receive 1024- to 1518-Byte Frame Register Definition

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

Table 13-52. 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 Register (TRMGV)

Figure 13-55 shows the TRMGV register.

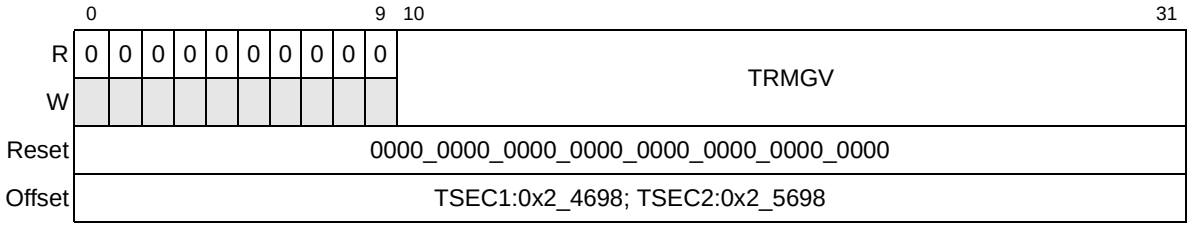


Figure 13-55. Transmit and Receive 1519- to 1522-Byte VLAN Frame Register Definition

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

Table 13-53. 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 Register (RBYT)

Figure 13-56 shows the RBYT register.

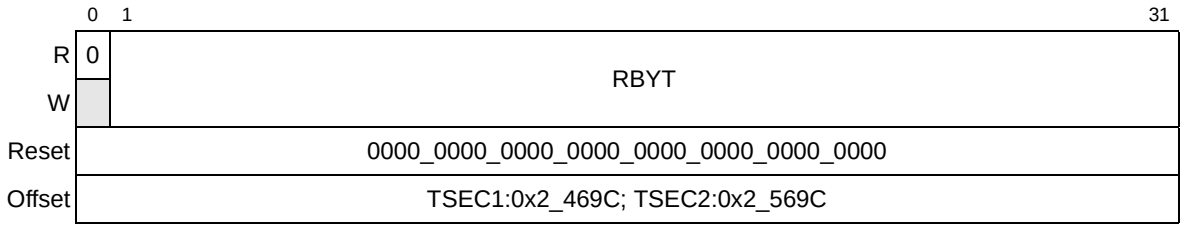


Figure 13-56. Receive Byte Counter Register Definition

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

Table 13-54. RBYT Field Descriptions

Bits	Name	Description
0	—	Reserved
1–31	RBYT	Receive byte counter. Increments by the byte count of frames received, including those in bad packets, excluding preamble and SFD but including FCS bytes.

13.5.3.7.9 Receive Packet Counter Register (RPKT)

Figure 13-57 shows the RPKT register.

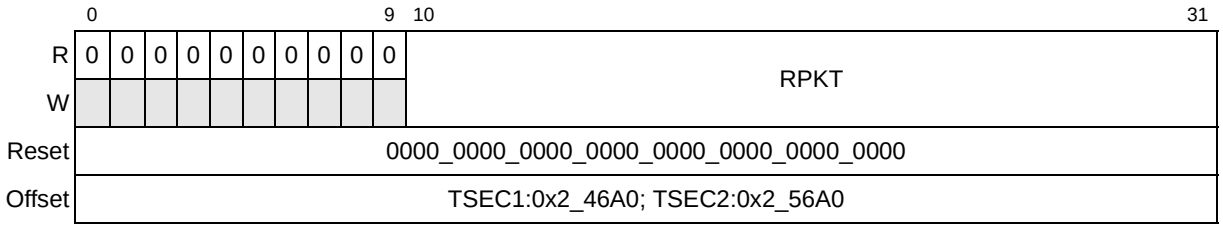


Figure 13-57. Receive Packet Counter Register Definition

Table 13-55 describes the fields of the RPKT register.

Table 13-55. 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 Register (RFCS)

Figure 13-58 shows the RFCS register.

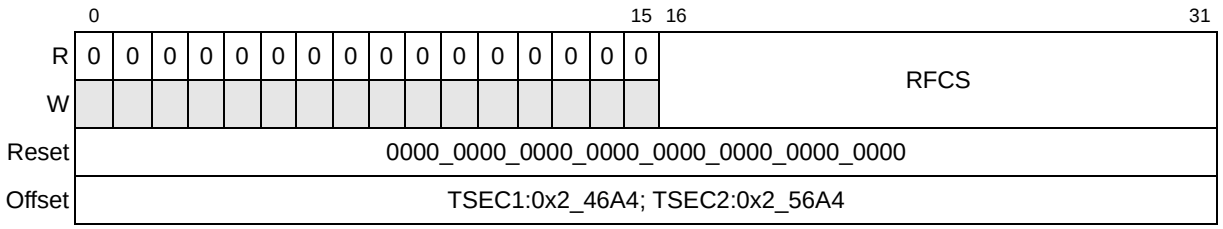


Figure 13-58. Receive FCS Error Counter Register Definition

Table 13-56 describes the fields of the RFCS register.

Table 13-56. 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 Register (RMCA)

Figure 13-59 shows the RMCA register.

	0									9	10	31
R	0	0	0	0	0	0	0	0	0	0	RMCA	
W												
Reset	0000_0000_0000_0000_0000_0000_0000_0000											
Offset	TSEC1:0x2_46A8; TSEC2:0x2_56A8											

Figure 13-59. Receive Multicast Packet Counter Register Definition

Table 13-57 describes the fields of the RMCA register.

Table 13-57. 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 Register (RBCA)

Figure 13-59 shows the RBCA register.

	0									9	10	31
R	0	0	0	0	0	0	0	0	0	0	RBCA	
W												
Reset	0000_0000_0000_0000_0000_0000_0000_0000											
Offset	TSEC1:0x2_46AC; TSEC2:0x2_56AC											

Figure 13-60. Receive Broadcast Packet Counter Register Definition

Table 13-58 describes the fields of the RBCA register.

Table 13-58. 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 Register (RXCF)

Figure 13-61 shows the RXCF register.

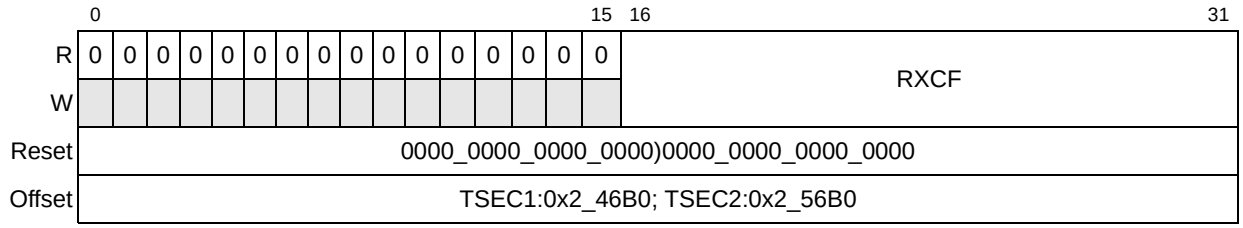


Figure 13-61. Receive Control Frame Packet Counter Register Definition

Table 13-59 describes the fields of the RXCF register.

Table 13-59. RXCF Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RXCF	Receive control frame packet counter. Increments for each MAC control frame received (both pause control frames and control frames unsupported by IEEE).

13.5.3.7.14 Receive Pause Frame Packet Counter Register (RXPF)

Figure 13-62 shows the RXPF register.

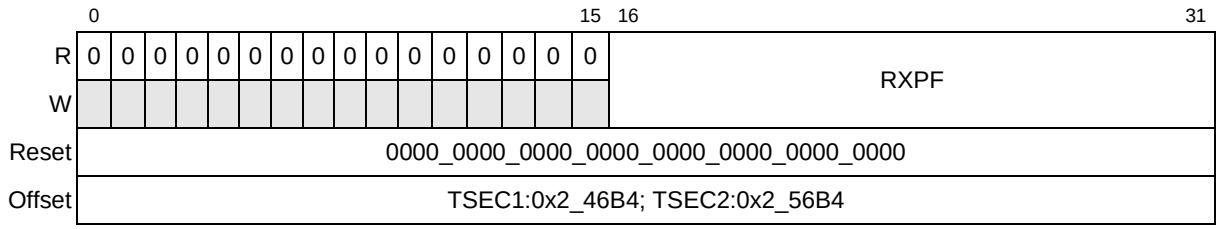


Figure 13-62. Receive Pause Frame Packet Counter Register Definition

Table 13-60 describes the fields of the RXPF register.

Table 13-60. 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 Register (RXUO)

Figure 13-63 shows the RXUO register.

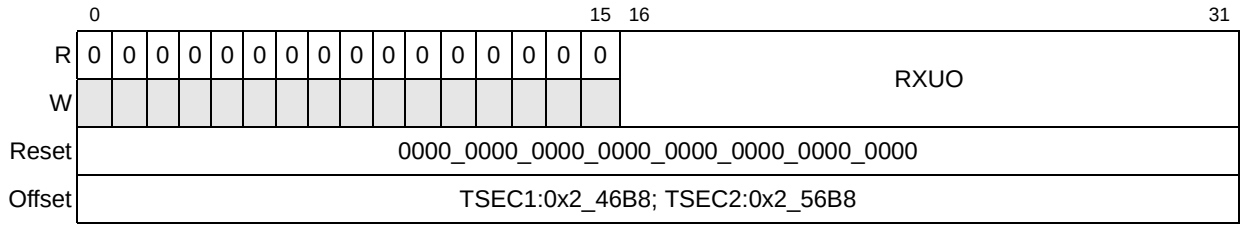


Figure 13-63. Receive Unknown Opcode Packet Counter Register Definition

Table 13-61 describes the fields of the RXUO register.

Table 13-61. 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 Register (RALN)

Figure 13-64 shows the RALN register.

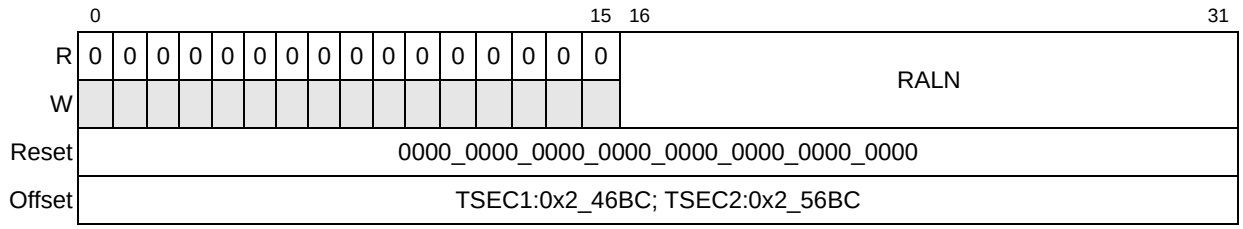


Figure 13-64. Receive Alignment Error Counter Register Definition

Table 13-62 describes the fields of the RALN register.

Table 13-62. 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 Register (RFLR)

Figure 13-65 shows the RFLR register.

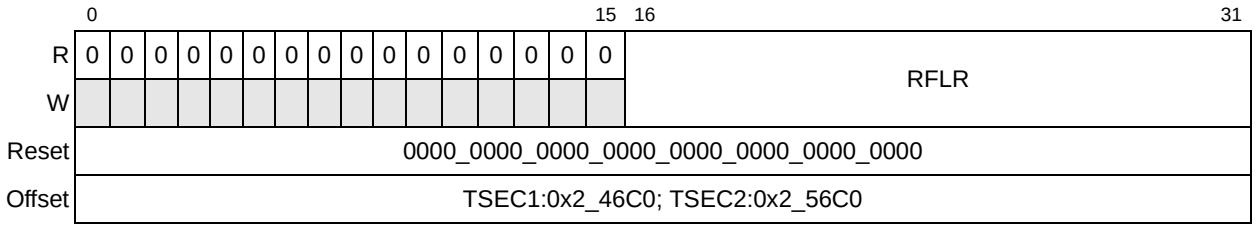


Figure 13-65. Receive Frame Length Error Counter Register Definition

Table 13-63 describes the fields of the RFLR register.

Table 13-63. 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 Register (RCDE)

Figure 13-66 shows the RCDE register.

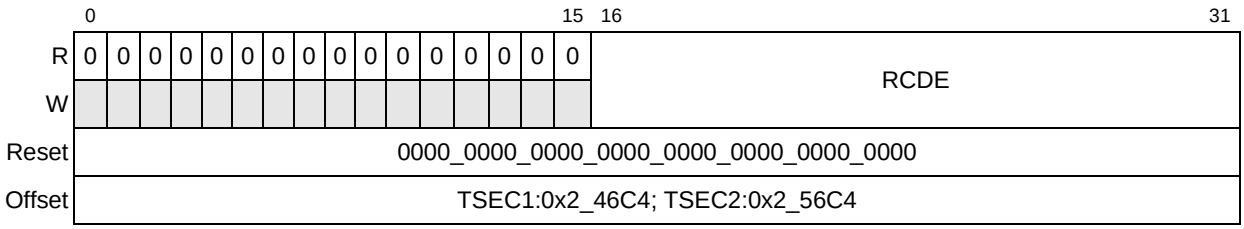


Figure 13-66. Receive Code Error Counter Register Definition

Table 13-64 describes the fields of the RCDE register.

Table 13-64. 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 Register (RCSE)

Figure 13-67 shows the RCSE register.

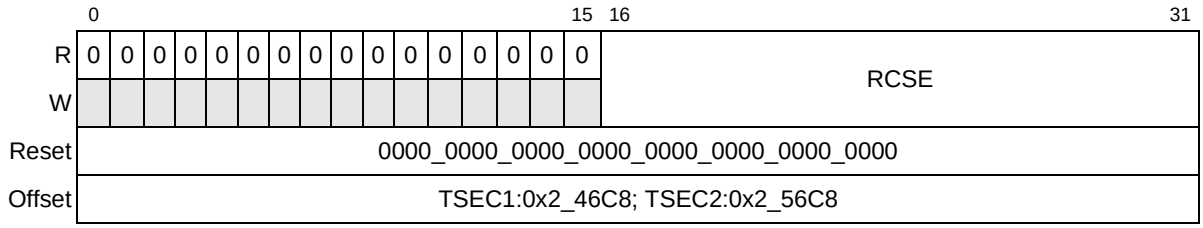


Figure 13-67. Receive Carrier Sense Error Counter Register Definition

Table 13-65 describes the fields of the RCSE register.

Table 13-65. 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 Register (RUND)

Figure 13-68 shows the RUND register.

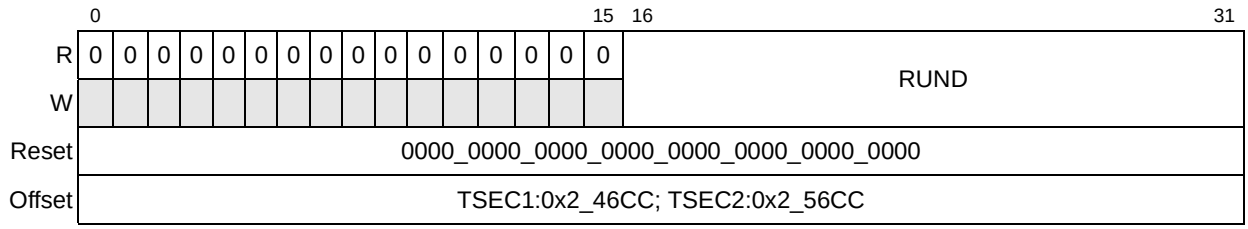


Figure 13-68. Receive Undersize Packet Counter Register Definition

Table 13-66 describes the fields of the RUND register.

Table 13-66. 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 Register (ROVR)

Figure 13-69 shows the ROVR register.

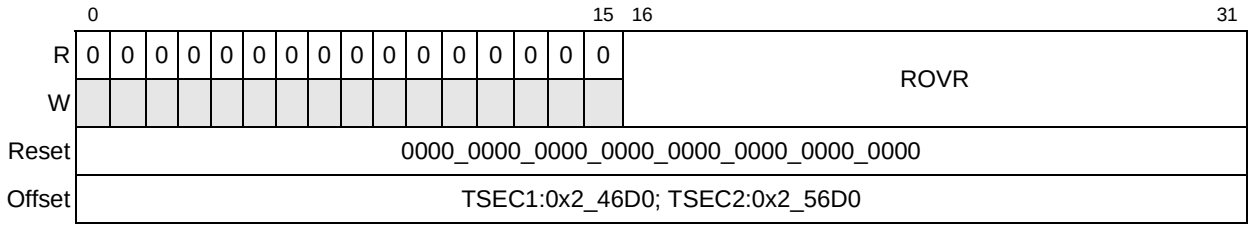


Figure 13-69. Receive Oversize Packet Counter Register Definition

Table 13-67 describes the fields of the ROVR register.

Table 13-67. 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 Register (RFRG)

Figure 13-70 shows the RFRG register.

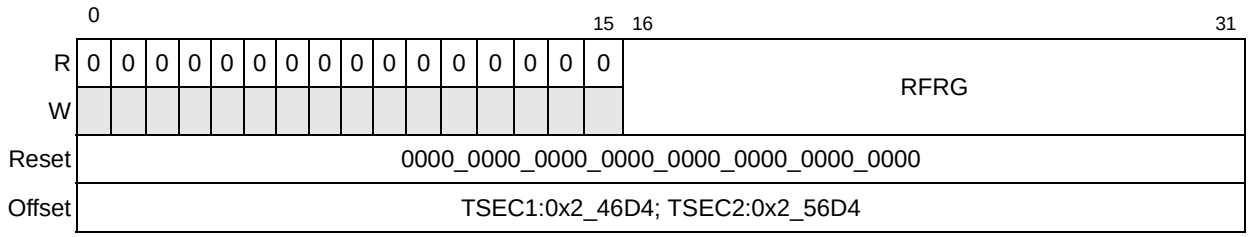


Figure 13-70. Receive Fragments Counter Register Definition

Table 13-68 describes the fields of the RFRG register.

Table 13-68. 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 Register (RJBR)

Figure 13-71 shows the RJBR register.

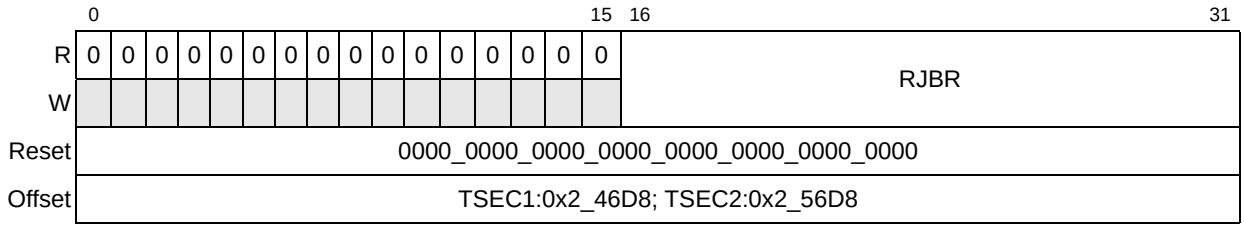


Figure 13-71. Receive Jabber Counter Register Definition

Table 13-69 describes the fields of the RJBR register.

Table 13-69. 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 Register (RDRP)

Figure 13-72 shows the RDRP register.

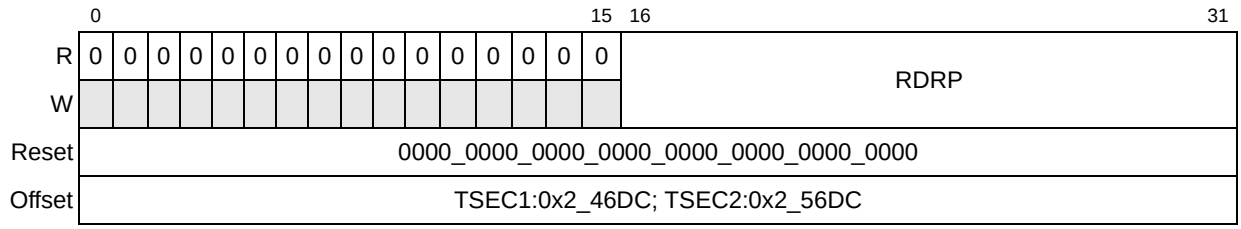


Figure 13-72. Receive Dropped Packet Counter Register Definition

Table 13-70 describes the fields of the RDRP register.

Table 13-70. 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 Register (TBYT)

Figure 13-73 shows the TBYT register.

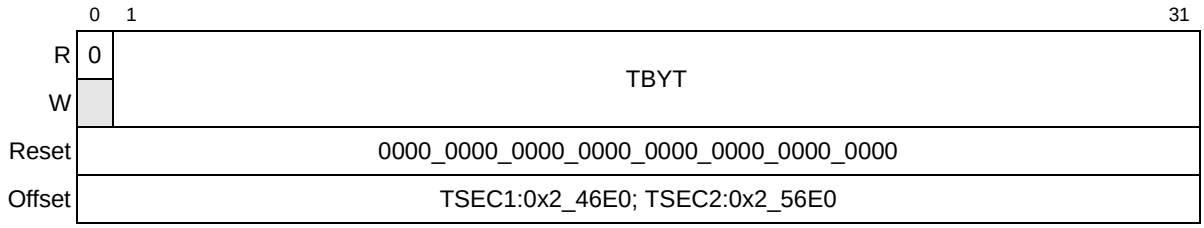


Figure 13-73. Transmit Byte Counter Register Definition

Table 13-71 describes the fields of the TBYT register.

Table 13-71. 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. This counter does not count if the frame is truncated.

13.5.3.7.26 Transmit Packet Counter Register (TPKT)

Figure 13-74 shows the TPKT register.

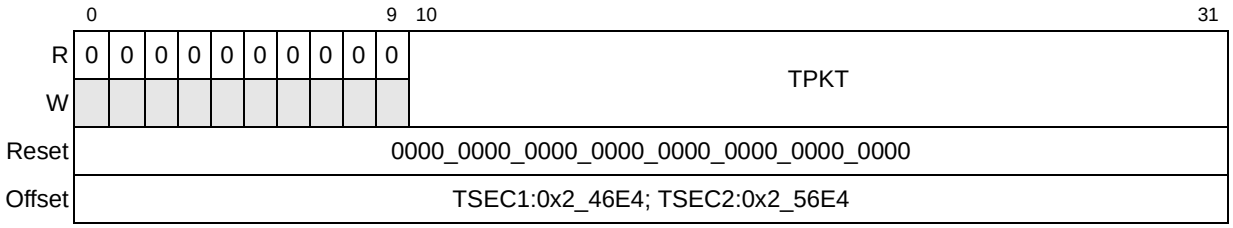


Figure 13-74. Transmit Packet Counter Register Definition

Table 13-72 describes the fields of the TPKT register.

Table 13-72. 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 Register (TMCA)

Figure 13-75 shows the TMCA register.

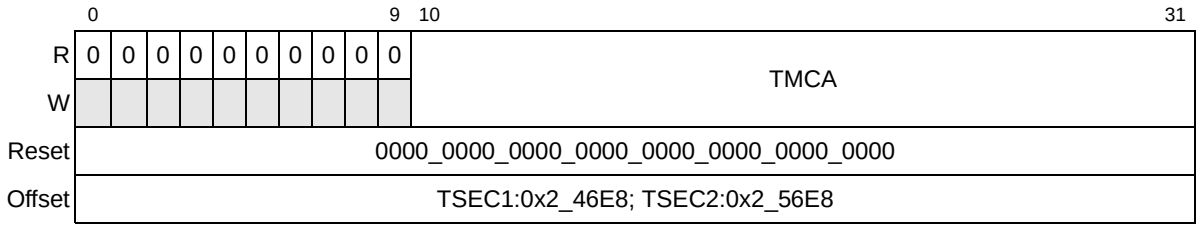


Figure 13-75. Transmit Multicast Packet Counter Register Definition

Table 13-73 describes the fields of the TMCA register.

Table 13-73. 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 Register (TBCA)

Figure 13-76 shows the TBCA register.

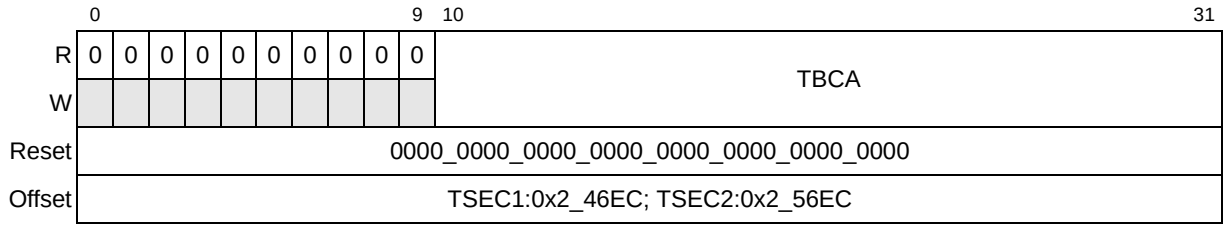


Figure 13-76. Transmit Broadcast Packet Counter Register Definition

Table 13-74 describes the fields of the TBCA register.

Table 13-74. 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 Register (TXPF)

Figure 13-77 shows the TXPF register.

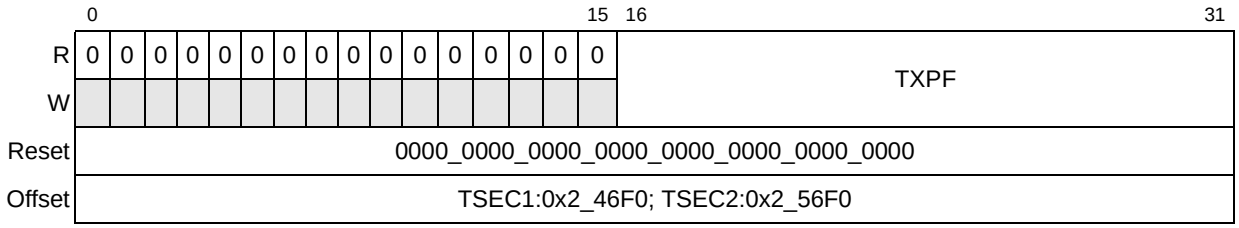


Figure 13-77. Transmit Pause Control Frame Counter Register Definition

Table 13-75 describes the fields of the TXPF register.

Table 13-75. 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 Register (TDFR)

Figure 13-78 shows the TDFR register.

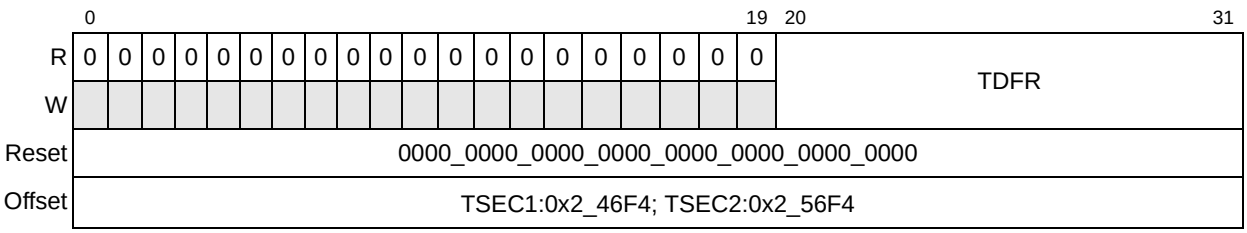


Figure 13-78. Transmit Deferral Packet Counter Register Definition

Table 13-76 describes the fields of the TDFR register.

Table 13-76. 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 Register (TEDF)

Figure 13-79 shows the TEDF register.

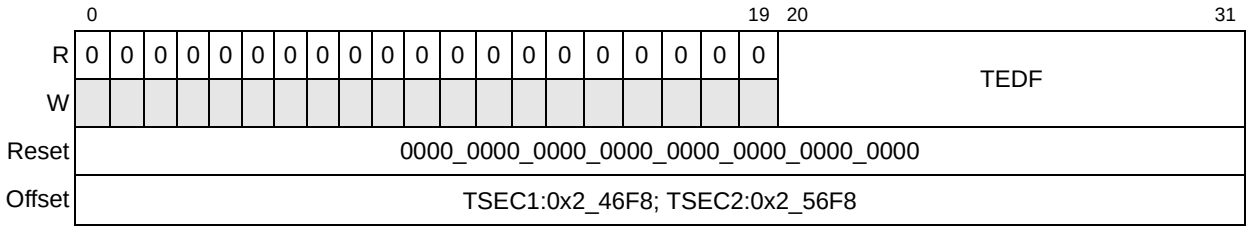


Figure 13-79. Transmit Excessive Deferral Packet Counter Register Definition

Table 13-77 describes the fields of the TEDF register.

Table 13-77. 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 Register (TSCL)

Figure 13-80 shows the TSCL register.

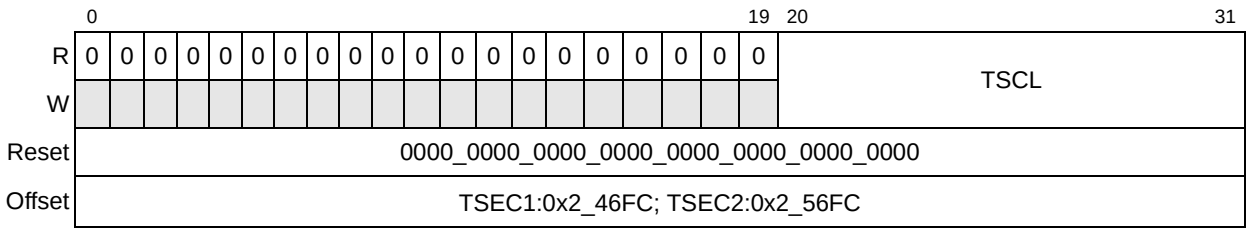


Figure 13-80. Transmit Single Collision Packet Counter Register Definition

Table 13-78 describes the fields of the TSCL register.

Table 13-78. 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 Register (TMCL)

Figure 13-81 shows the TMCL register.

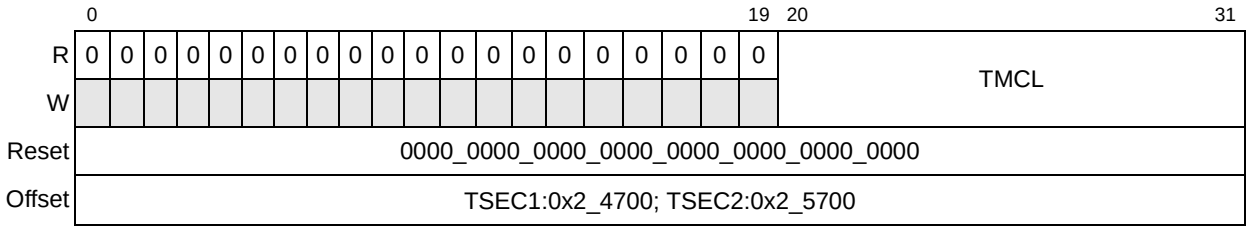


Figure 13-81. Transmit Multiple Collision Packet Counter Register Definition

Table 13-79 describes the fields of the TMCL register.

Table 13-79. TMCL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20–31	TMCL	Transmit multiple collision packet counter. Increments for each frame transmitted which experienced 2-15 collisions (including any late collisions) during transmission as defined using the Half_Duplex[RETRANSMISSION MAXIMUM] field.

13.5.3.7.34 Transmit Late Collision Packet Counter Register (TLCL)

Figure 13-82 shows the TLCL register.

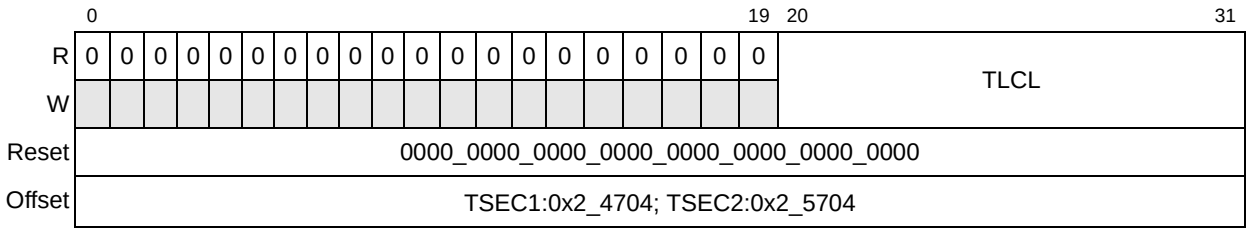


Figure 13-82. Transmit Late Collision Packet Counter Register Definition

Table 13-80 describes the fields of the TLCL register.

Table 13-80. 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 Register (TXCL)

Figure 13-83 shows the TXCL register.

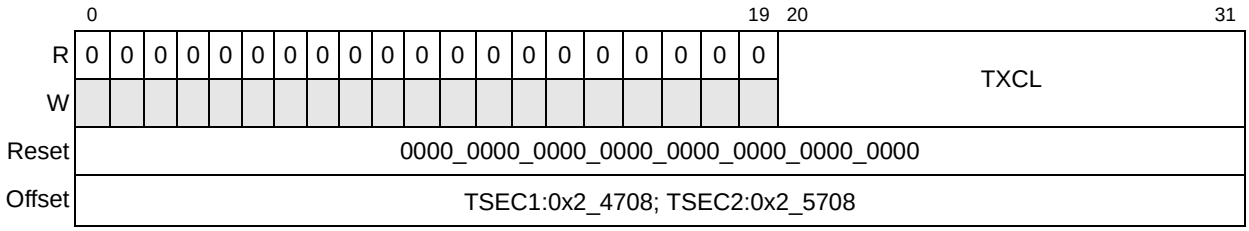


Figure 13-83. Transmit Excessive Collision Packet Counter Register Definition

Table 13-81. 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 Register (TNCL)

Figure 13-84 shows the TNCL register.

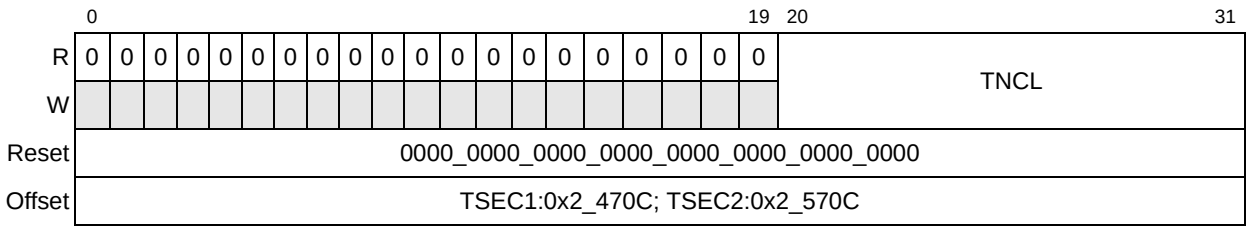


Figure 13-84. Transmit Total Collision Counter Register Definition

Table 13-82 describes the fields of the TNCL register.

Table 13-82. 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 Register (TDRP)

Figure 13-85 shows the TDRP register.

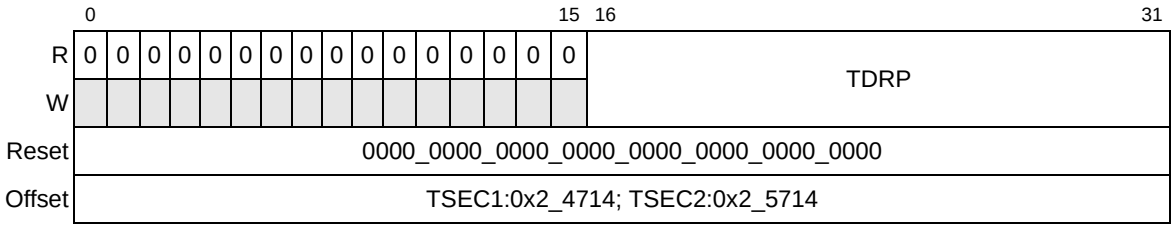


Figure 13-85. Transmit Drop Frame Counter Register Definition

Table 13-83 describes the fields of the TDRP register.

Table 13-83. TDRP Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TDRP	Transmit drop frame counter. Increments each time a memory error or an underrun has occurred.

13.5.3.7.38 Transmit Jabber Frame Counter Register (TJBR)

Figure 13-86 shows the TJBR register.

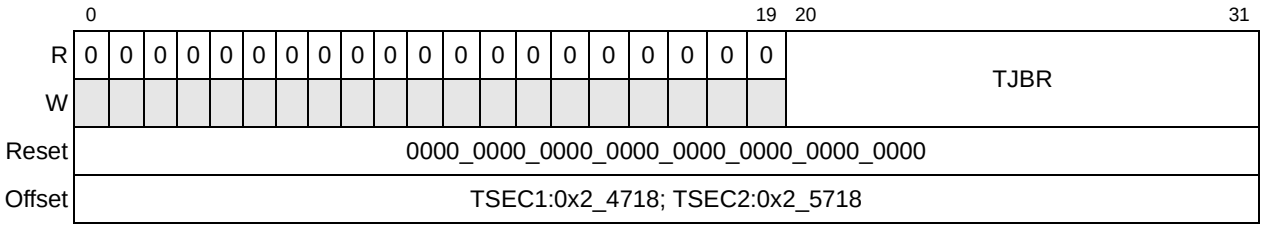


Figure 13-86. Transmit Jabber Frame Counter Register Definition

Table 13-84 describes the fields of the TJBR register.

Table 13-84. 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 Register (TFCS)

Figure 13-87 shows the TFCS register.

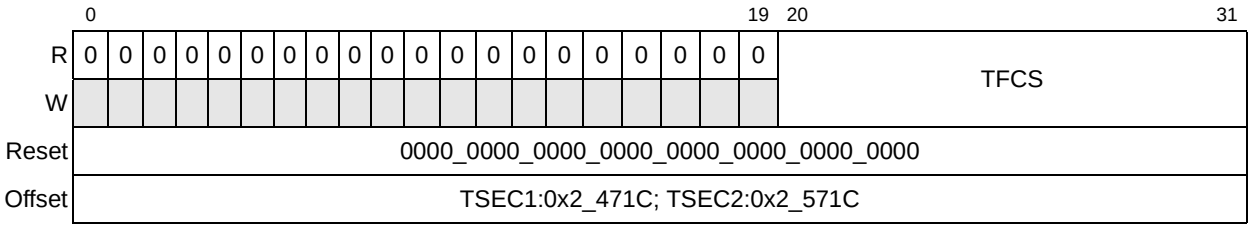


Figure 13-87. Transmit FCS Error Counter Register Definition

Table 13-85 describes the fields of the TFCS register.

Table 13-85. 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 Register (TXCF)

Figure 13-88 shows the TXCF register.

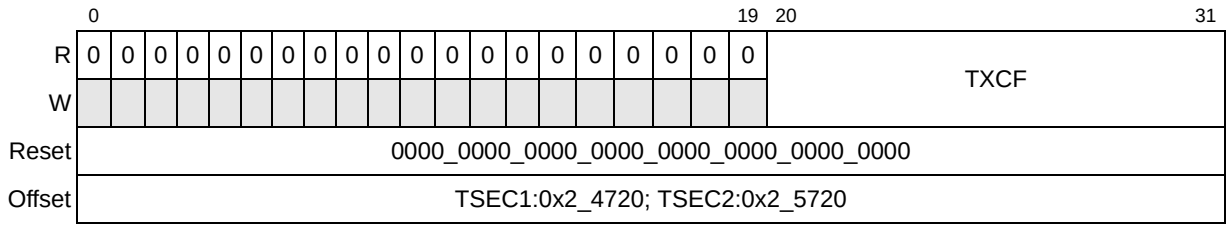


Figure 13-88. Transmit Control Frame Counter Register Definition

Table 13-86 describes the fields of the TXCF register.

Table 13-86. 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 Register (TOVR)

Figure 13-89 shows the TOVR register.

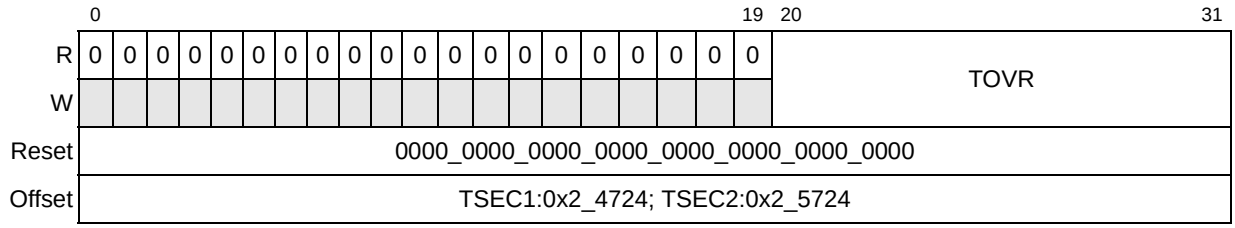


Figure 13-89. Transmit Oversized Frame Counter Register Definition

Table 13-87 describes the fields of the TOVR register.

Table 13-87. 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 Register (TUND)

Figure 13-90 shows the TUND register.

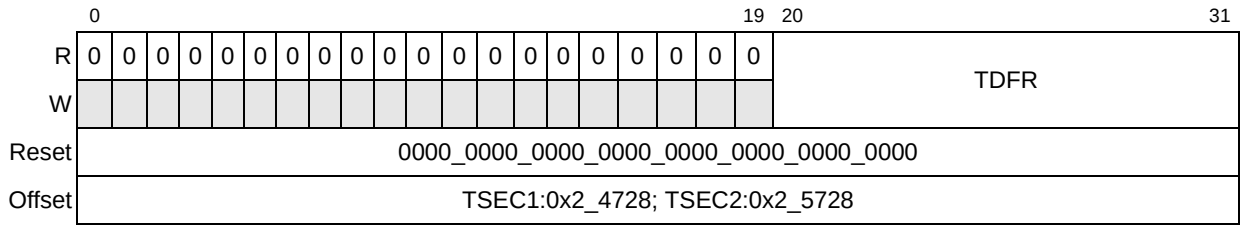


Figure 13-90. Transmit Undersize Frame Counter Register Definition

Table 13-88 describes the fields of the TUND register.

Table 13-88. 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 Register (TFRG)

Figure 13-91 shows the TFRG register.

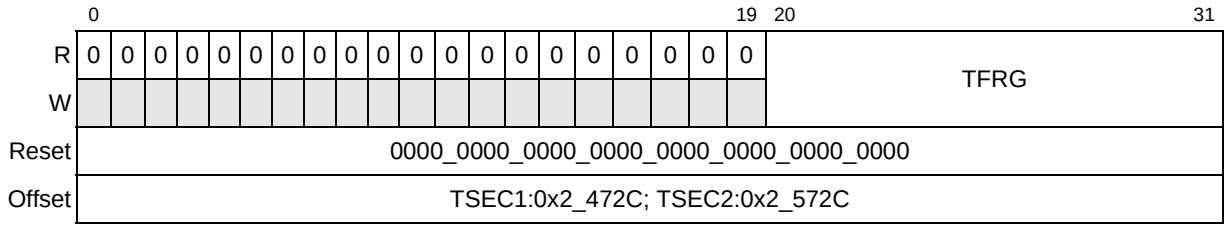


Figure 13-91. Transmit Fragment Counter Register Definition

Table 13-89 describes the fields of the TFRG register.

Table 13-89. 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 1 (CAR1)

Carry register bits are cleared when written with a one. Figure 13-92 shows the CAR1 register.

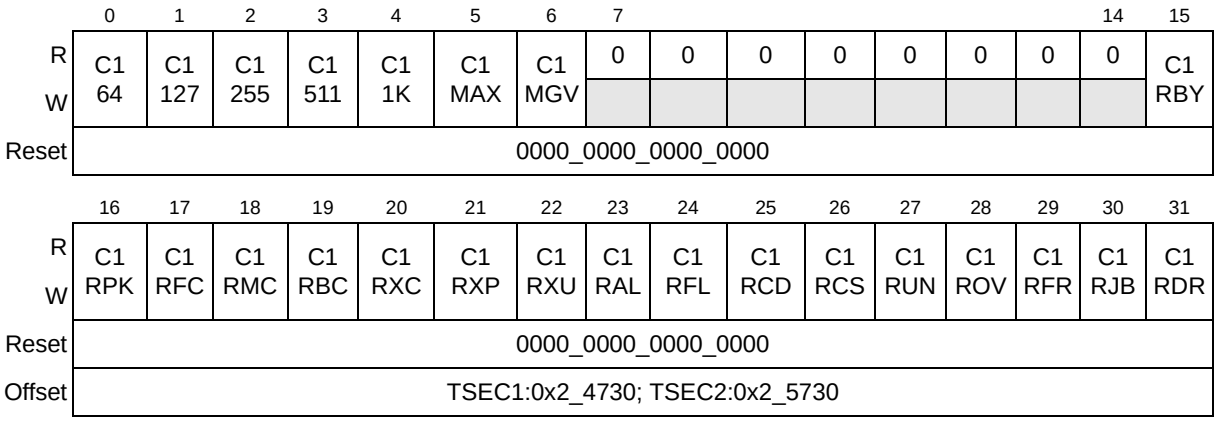


Figure 13-92. Carry Register 1 (CAR1) Register Definition

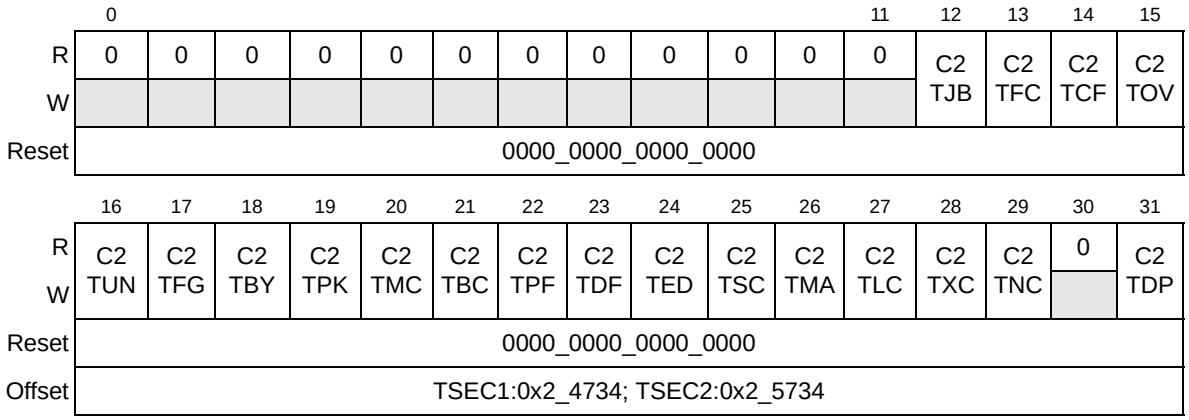
Table 13-90 describes the fields of the CAR1 register.

Table 13-90. 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
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 2 (CAR2)

Carry register bits are cleared when written with a one. Figure 13-93 shows the CAR2 register.



	0											11	12	13	14	15
R	0	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	0	C2 TDP
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_4734; TSEC2:0x2_5734															

Figure 13-93. Carry Register 2 (CAR2) Register Definition

Table 13-91 describes the fields of the CAR2 register.

Table 13-91. CAR2 Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12	C2TJB	Carry register 2 TJBR counter carry bit
13	C2TFC	Carry register 2 TFCS counter carry bit
14	C2TCF	Carry register 2 TXCF counter carry bit
15	C2TOV	Carry register 2 TOVR counter carry bit
16	C2TUN	Carry register 2 TUND counter carry bit
17	C2TFG	Carry register 2 TFRG counter carry bit
18	C2TBY	Carry register 2 TBYT counter carry bit
19	C2TPK	Carry register 2 TPKT counter carry bit
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

Table 13-91. CAR2 Field Descriptions (continued)

Bits	Name	Description
29	C2TNC	Carry register 2 TNCL counter carry bit
30	—	Reserved
31	C2TDP	Carry register 2 TDRP counter carry bit

13.5.3.7.46 Carry Mask Register 1 (CAM1)

As long as one of the below mask bits is cleared, the corresponding interrupt bit is allowed to cause interrupt indications on output CARRY. These bits all default to a set state. Figure 13-94 shows the CAM1 register.

	0	1	2	3	4	5	6	7							14	15
R	M1 64	M1 127	M1 255	M1 511	M1 1K	M1 MAX	M1 MGV	0	0	0	0	0	0	0	0	M1 RBY
W																
Reset	1111_1110_0000_0001															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M1 RPK	M1 RFC	M1 RMC	M1 RBC	M1 RXC	M1 RXP	M1 RXU	M1 RAL	M1 RFL	M1 RCD	M1 RCS	M1 RUN	M1 ROV	M1 RFR	M1 RJB	M1 RDR
W																
Reset	1111_1111_1111_1111															
Offset	TSEC1:0x2_4738; TSEC2:0x2_5738															

Figure 13-94. Carry Mask Register 1 (CAM1) Register Definition

Table 13-92 describes the fields of the CAM1 register.

Table 13-92. 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
15	M1RBY	Mask register 1 RBYT counter carry bit mask
16	M1RPK	Mask register 1 RPKT counter carry bit mask

Table 13-92. CAM1 Field Descriptions (continued)

Bits	Name	Description
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 2 (CAM2)

As long as one of the below mask bits is cleared, the corresponding interrupt bit is allowed to cause interrupt indications on output CARRY. These bits default to a set state. Figure 13-95 shows the CAM2 register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	M2 TJB	M2 TFC	M2 TCF	M2 TOV
W																
Reset	0000_0000_0000_1111															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M2 TUN	M2 TFG	M2 TBY	M2 TPK	M2 TMC	M2 TBC	M2 TPF	M2 TDF	M2 TED	M2 TSC	M2 TMA	M2 TLC	M2 TXC	M2 TNC	0	M2 TDP
W																
Reset	1111_1111_1111_1111															
Offset	TSEC1:0x2_473C; TSEC2:0x2_573C															

Figure 13-95. Carry Mask Register 2 (CAM2) Register Definition

Table 13-93 describes the fields of the CAM2 register.

Table 13-93. CAM2 Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12	M2TJB	Mask register 2 TJBR counter carry bit mask
13	M2TFC	Mask register 2 TFCS counter carry bit mask
14	M2TCF	Mask register 2 TXCF counter carry bit mask
15	M2TOV	Mask register 2 TOVR counter carry bit mask
16	M2TUN	Mask register 2 TUND counter carry bit mask
17	M2TFG	Mask register 2 TFRG counter carry bit mask
18	M2TBY	Mask register 2 TBYT counter carry bit mask
19	M2TPK	Mask register 2 TPKT counter carry bit mask
20	M2TMC	Mask register 2 TMCA counter carry bit mask
21	M2TBC	Mask register 2 TBCA counter carry bit mask
22	M2TPF	Mask register 2 TXPF counter carry bit mask
23	M2TDF	Mask register 2 TDFR counter carry bit mask
24	M2TED	Mask register 2 TEDF counter carry bit mask
25	M2TSC	Mask register 2 TSCL counter carry bit mask
26	M2TMA	Mask register 2 TMCL counter carry bit mask
27	M2TLC	Mask register 2 TLCL counter carry bit mask
28	M2TXC	Mask register 2 TXCL counter carry bit mask
29	M2TNC	Mask register 2 TNCL counter carry bit mask
30	—	Reserved
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 must 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* registers, shown in Figure 13-96, are written by the user. These registers represent 256 entries of the individual (unicast) address hash table used in the address recognition process. When the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 high-order bits (0–7) of the CRC remainder are mapped to one of the 256 entries. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry.

	0	31
R W	IADDR*n*	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
Offset	TSEC1:0x2_4800, 0x2_4804, 0x2_4808, 0x2_480C, 0x2_4810, 0x2_4814, 0x2_4818, 0x2_481C TSEC2:0x2_5800, 0x2_5804, 0x2_5808, 0x2_580C, 0x2_5810, 0x2_5814, 0x2_5818, 0x2_581C	

Figure 13-96. IADDR*n* Register Definition

Table 13-94 describes the field of the IADDR*n* registers.

Table 13-94. IADDR*n* Field Description

Bits	Name	Description
0–31	IADDR*n*	Represents the 32-bit value associated with the corresponding register. IADDR0 contains the high-order 32 bits of the 256-entry hash table and IADDR7 represents the low-order 32 bits. For instance, the MSB of IADDR0 correlates to entry 0 and the LSB of IADDR7 correlates to entry 255.

13.5.3.8.2 Group Address Registers 0–7 (GADDR*n*)

The GADDR*n* registers are written by the user. Together these registers represent 256 entries of the group (multicast) address hash table used in the address recognition process. While the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 bits of the CRC remainder is mapped to one of the 256 entries. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs if the DA CRC result points to an enabled hash entry. Figure 13-97 shows the GADDR*n* registers.

	0	31
R W	GADDR*n*	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
Offset	TSEC1:0x2_4880, 0x2_4884, 0x2_4888, 0x2_488C, 0x2_4890, 0x2_4894, 0x2_4898, 0x2_489C TSEC2:0x2_5880, 0x2_5884, 0x2_5888, 0x2_588C, 0x2_5890, 0x2_5894, 0x2_5898, 0x2_589C	

Figure 13-97. GADDR*n* Register Definition

Table 13-95 describes GADDR*n*.

Table 13-95. 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. For instance, the MSB of GADDR0 correlates to entry 0 and the LSB of GADDR7 correlates to entry 255.

13.5.3.9 Attribute Registers

This section describes the two TSEC frame attribute registers.

13.5.3.9.1 Attribute Register (ATTR)

The attribute register, shown in Figure 13-98, defines attributes and transaction types used to access buffer descriptors, to write receive data, and to read transmit data. Snoop enable attributes may be set for reading buffer descriptors and for reading transmit data. Buffer descriptors may be written with attributes that cause allocation into the L2 cache with or without line locking. Similarly, sections of a receive frame header may have attributes attached that cause allocation and locking in the L2 cache. This process of specifying a region of each frame to stash into the L2 cache is referred to as extraction, which is specified in conjunction with ATTRELI. ATTR[ELCWT] only has meaning if ATTRELI[EL] is non-zero.

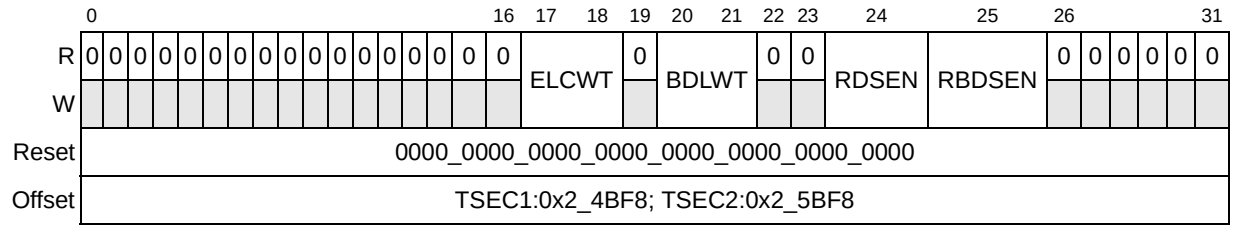


Figure 13-98. ATTR Register Definition

Table 13-96 describes the fields of the ATTR register.

Table 13-96. ATTR Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17–18	ELCWT	Extracted L2 cache write type. Specifies the write transaction type to perform for the extracted data. This occurs only if ATTRELI[EL] is non-zero. For maximum performance, it is recommended that if ELCWT is set to allocate, BDLWT must also be set to allocate. Writes to cache are always performed with snoop. This setting overrides the RDSEN bit setting. 00 No allocation performed 01 Reserved, no extraction occurs 10 Allocate L2 cache line 11 Allocate and lock L2 cache line

Table 13-96. ATTR Field Descriptions (continued)

Bits	Name	Description
19	—	Reserved
20–21	BDLWT	Buffer descriptor L2 cache write type. Specifies the write transaction type to perform for the buffer descriptor for a receive frame. Writes to cache are always performed with snoop. 00 No allocation performed. This setting overrides the RBDSEN bit setting. 01 Reserved 10 Allocate L2 cache line 11 Allocate and lock L2 cache line
22–23	—	Reserved
24	RDSEN	Rx data snoop enable. This bit is superseded by the ELCWT settings. 0 Disables snooping of all receive frames data to memory unless ELCWT specifies L2 allocation 1 Enables snooping of all receive frames data to memory
25	RBDSEN	RxBD snoop enable. This bit is superseded by the BDLWT settings. 0 Disables snooping of all receive BD memory accesses unless BDLWT specifies L2 allocation 1 Enables snooping of all receive BD memory accesses
26–31	—	Reserved

13.5.3.9.2 Attribute Extract Length and Extract Index Register (ATTRELI)

The ATTRELI registers are written by the user to specify the extract index and extract length. Figure 13-99 shows the ATTRELI register.

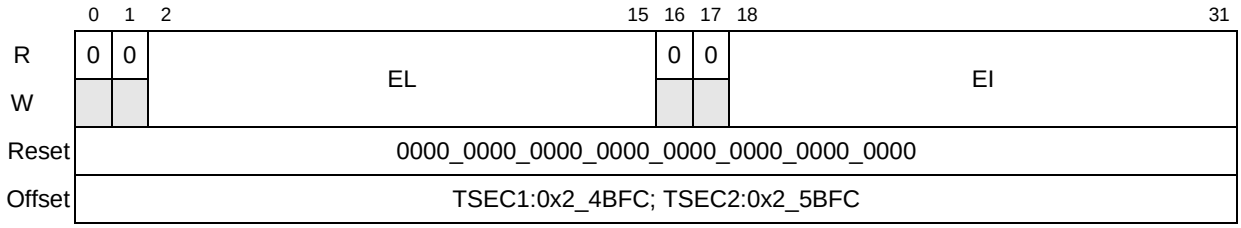


Figure 13-99. ATTRELI Register Definition

Table 13-97 describes the fields of the ATTRELI register.

Table 13-97. 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 TBI registers are 16 bits wide and are accessed at the offset of the TBI physical address. The TBI physical address of the TSEC is stored in TBIPA. Writing to the TBI registers is 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 MIIMADD[Register Address] to the address offset that corresponds to the desired register (see Table 13-98), 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 13-98.

Note 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 13-98. TBI MII Register Set

Offset	Name	Access	Size	Section/Page
TEN-BIT INTERFACE (TBI) REGISTERS				13.5.4/13-91
0x00	Control (CR)	R/W[1]	16 bits	13.5.4.2/13-92
0x01	Status (SR)	R, LH, LL	16 bits	13.5.4.3/13-93
0x02–0x03	Reserved	R	2 bytes	—
0x04	AN advertisement (ANA)	RW, R	16 bits	13.5.4.3/13-93
0x05	AN link partner base page ability (ANLPBPA)	R	16 bits	13.5.4.5/13-96
0x06	AN expansion (ANEX)	R, LH	16 bits	13.5.4.6/13-98
0x07	AN next page transmit (ANNPT)	R/W, R	16 bits	13.5.4.7/13-98
0x08	AN link partner ability next page (ANLPANP)	R	16 bits	13.5.4.8/13-99
0x0F	Extended status (EXST)	R	16 bits	13.5.4.9/13-100
0x10	Jitter diagnostics (JD)	R/W	16 bits	13.5.4.10/13-101
0x11	TBI control (TBICON)	R/W	16 bits	13.5.4.11/13-102

[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 13-100 shows the CR register.

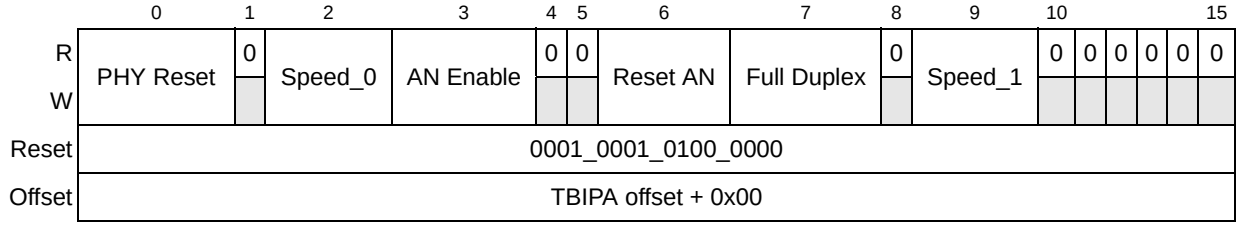


Figure 13-100. Control Register Definition

Table 13-99 describes the fields of the CR register.

Table 13-99. CR Field Descriptions

Bits	Name	Description
0	PHY Reset	PHY reset. This bit is cleared by default. This bit is self-clearing. 0 Normal operation. 1 The control and status registers are returned to their default value. This in turn may change the internal state of the TBI and its link partner.
1	—	Reserved
2	Speed_0	Speed selection. This bit defaults to a cleared state and must always be cleared, which corresponds to 1-Gbps speed.
3	AN Enable	Auto-negotiation enable. This bit is set by default. 0 The values programmed in bits 2, 7 and 9 determine the operating condition of the link. 1 Auto-negotiation process enabled
4–5	—	Reserved
6	Reset AN	Reset auto-negotiation. This bit is cleared by default and is self-clearing. 0 Normal operation 1 The auto-negotiation process restarts. This action is only available if auto-negotiation is enabled.
7	Full Duplex	Duplex mode. This bit is set by default. 0 Half-duplex operation 1 Full-duplex operation
8	—	Reserved, must be cleared.

Table 13-99. CR Field Descriptions (continued)

Bits	Name	Description
9	Speed_1	Speed selection. Defaults to a set state and must always be set, which corresponds to 1 Gbps **Maximum Operating Speed** / **Bit 2** / **Bit 9** Reserved / 0 / 0 Reserved / 1 / 0 1 Gbps / 0 / 1 Reserved / 1 / 1
10–15	—	Reserved

13.5.4.3 Status Register (SR)

Figure 13-101 shows 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	TBIPA offset + 0x01															

Figure 13-101. Status Register Definition

Table 13-100 describes the fields of the SR register.

Table 13-100. SR Descriptions

Bits	Name	Description
0–6	—	Reserved, must be cleared.
7	Extend Status	Indicates that PHY status information is also contained in the Extended Status Register. Returns 1 on read. This bit is read-only.
8	—	Reserved, must 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 SR clears this bit. 0 Normal operation 1 A remote fault condition was detected. Latches high for software to detect the condition.

Table 13-100. SR Descriptions (continued)

Bits	Name	Description
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. Latches low allowing for software polling to detect a failure condition. 1 A valid link is established.
14	—	Reserved, must 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 13-102 shows the ANA register.

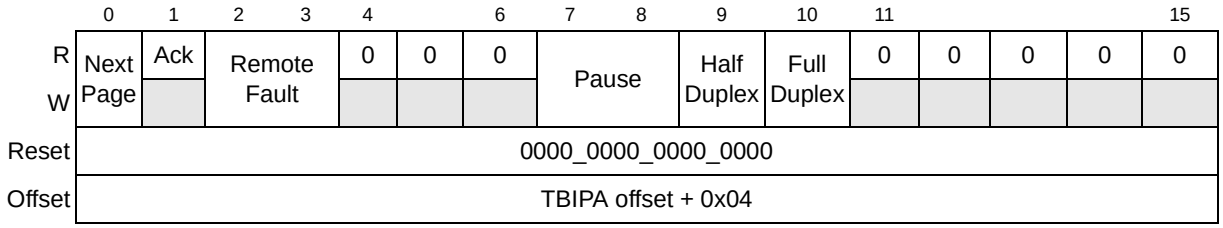


Figure 13-102. AN Advertisement Register Definition

Table 13-101 describes the fields of the ANA register.

Table 13-101. 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 of the *IEEE 802.3 specification*.
1	Ack	Acknowledge. Write 0, ignore on read.This bit is read-only. (Reserved)

Table 13-101. ANA Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>2–3</td><td>Remote Fault</td><td>The local device’s remote fault condition is encoded in bits 2 and 3 of the base page. Values are shown in the following table. The default value is 00. Indicate a fault by setting a non-zero remote fault encoding and re-negotiating.
<table>
<tr><th>RF1 bit[3]</th><th>RF2 bit[2]</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>No error, link OK</td></tr>
<tr><td>0</td><td>1</td><td>Offline</td></tr>
<tr><td>1</td><td>0</td><td>Link_Failure</td></tr>
<tr><td>1</td><td>1</td><td>Auto-Negotiation_Error</td></tr>
</table></td></tr>
<tr><td>4–6</td><td>—</td><td>Reserved, must be cleared.</td></tr>
<tr><td>7–8</td><td>Pause</td><td>The local device’s PAUSE capability is encoded in bits 7 and 8, and the decodes are shown in the following table. For priority resolution information consult Table 13-102.
<table>
<tr><th>PAUSE bit[8]</th><th>ASM_DIR bit[7]</th><th>Capability</th></tr>
<tr><td>0</td><td>0</td><td>No PAUSE</td></tr>
<tr><td>0</td><td>1</td><td>Asymmetric PAUSE toward link partner</td></tr>
<tr><td>1</td><td>0</td><td>Symmetric PAUSE</td></tr>
<tr><td>1</td><td>1</td><td>Both symmetric PAUSE and
Asymmetric PAUSE toward local device</td></tr>
</table></td></tr>
<tr><td>9</td><td>Half Duplex</td><td>Half-duplex capability
0 Designates local device as not capable of half-duplex operation.
1 Designates local device as capable of half-duplex operation.</td></tr>
<tr><td>10</td><td>Full Duplex</td><td>Full-duplex capability
0 Designates the local device as not capable of full-duplex operation.
1 Designates the local device as capable of full-duplex operation.</td></tr>
<tr><td>11–15</td><td>—</td><td>Reserved, must be cleared.</td></tr>
</table>

Table 13-102 describes the resolution of pause priority.

Table 13-102. 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 13-103 shows the ANLPBPA register.

	0	1	2	3	4		6	7	8	9	10	11				15
R	Next Page	Ack	Remote Fault		0	0	0	Pause		Half Duplex	Full Duplex	0	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	TBIPA offset + 0x05															

Figure 13-103. AN Link Partner Base Page Ability Register Definition

Table 13-103 describes the fields of the ANLPBPA register.

Table 13-103. 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, must 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 the IEEE 802.3 specification. 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, must be cleared.</td></tr>
</table>

13.5.4.6 AN Expansion Register (ANEX)

Figure 13-104 shows the ANEX register.

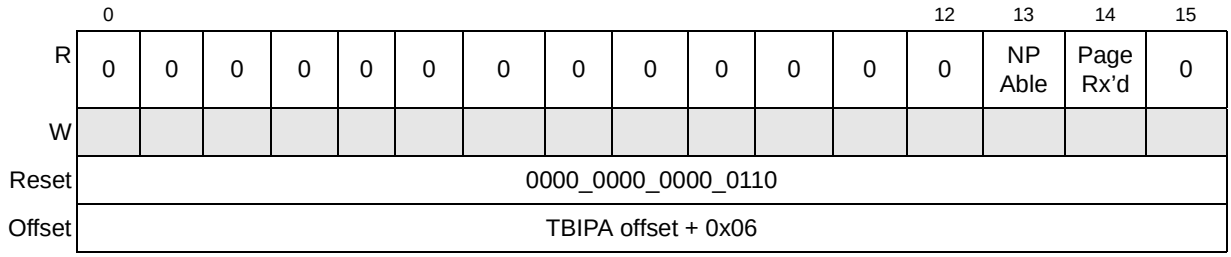


Figure 13-104. AN Expansion Register Definition

Table 13-104 describes the fields of the ANEX register.

Table 13-104. ANEX Field Descriptions

Bits	Name	Description
0–12	—	Reserved, must 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, must be cleared.

13.5.4.7 AN Next Page Transmit Register (ANNPT)

Figure 13-105 shows the ANNPT register.

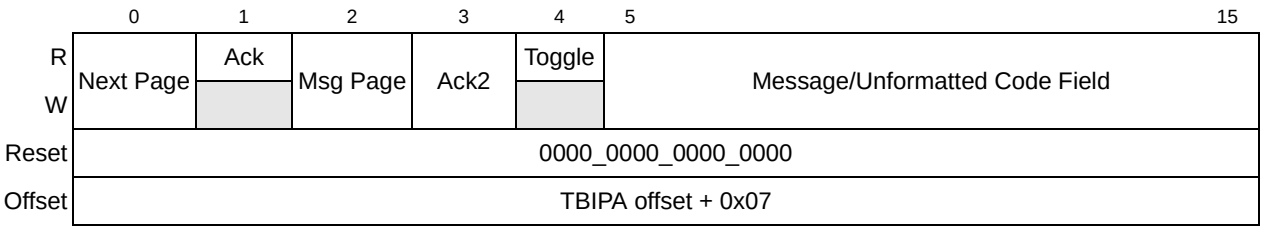


Figure 13-105. AN Next Page Transmit Register Definition

Table 13-105 describes the fields of the ANNPT register.

Table 13-105. 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	Acknowledge. 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 set 1 Toggle bit of the previously-exchanged link code word was clear
5–15	Message/ Unformatted 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 13-106 shows the ANLPANP register.

	0	1	2	3	4	5										15
R	Next Page	Ack	Msg Page	Ack2	Toggle	Message/Unformatted Code Field										
W																
Reset	0000_0000_0000_0000															
Offset	TBIPA offset + 0x08															

Figure 13-106. AN Link Partner Ability Next Page Register Definition

Table 13-106 describes the ANLPANP fields.

Table 13-106. 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	Acknowledge. Ignore on read. [Reference MII bit 8.14] This bit is read-only.

Table 13-106. ANLPANP Field Descriptions (continued)

Bits	Name	Description
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 set 1 Toggle bit of the previously-exchanged link code word was clear
5–15	Message/ Unformatted 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 13-107 shows 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	TBIPA offset + 0x0F															

Figure 13-107. Extended Status Register Definition

Table 13-107 describes the fields of the EXST register.

Table 13-107. 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 operate 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 operate 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 operate in 1000BASE-T full-duplex mode 1 PHY can operate in 1000BASE-T full-duplex mode

Table 13-107. EXST Field Descriptions (continued)

Bits	Name	Description
3	1000T Half	1000T half-duplex capability. Returns 0 on read. This bit is read-only. 0 PHY cannot operate in 1000BASE-T half-duplex mode 1 PHY can operate in 1000BASE-T half-duplex mode
4–15	—	Reserve

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. It may be wise to auto-negotiate and advertise a remote fault, signaling offline, prior to beginning the test patterns. Figure 13-108 shows the JD register.

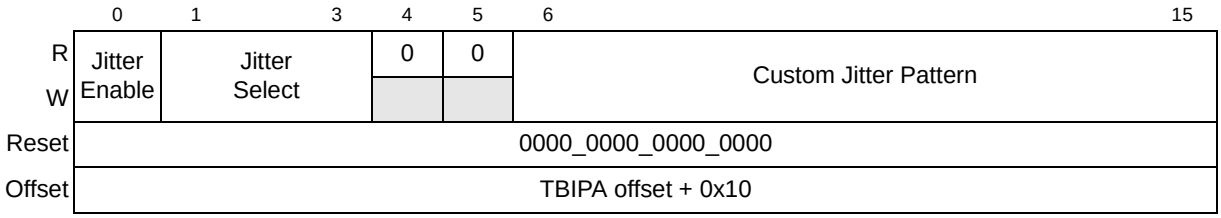


Figure 13-108. Jitter Diagnostics Register Definition

Table 13-108 describes the fields of the JD register.

Table 13-108. JD Field Description

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>0</td><td>Jitter Enable</td><td>Jitter enable. This bit is cleared by default.
0 Normal transmit operation
1 Enable the TBI to transmit the jitter test patterns defined in annex 36A of the IEEE 802.3 specification.</td></tr>
<tr><td>1–3</td><td>Jitter Select</td><td>Selects the jitter pattern to be transmitted in diagnostics mode. Encoding of this field is shown in the following table. Default is 0
<table>
<tr><th>Jitter Pattern Select</th><th>bit[1]</th><th>bit[2]</th><th>bit[3]</th></tr>
<tr><td>User defined uses custom jitter pattern, bits 6–15</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>High frequency (±D21.5)
10101010101010101010101010101010101010…</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>Mixed frequency (±K28.5)
11111010110000010100111110101100000101 00…</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>Low frequency
11111000001111100000111110000011111100000…</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>Complex pattern (10’h17c,10’h0c9,10’h0e5,10’h2a3, 10’h17c,...)</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>Square Wave (-K28.7)
00111110000011111000001111100000111110000…</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>Reserved</td><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>4–5</td><td>—</td><td>Reserved</td></tr>
<tr><td>6–15</td><td>Custom Jitter Pattern</td><td>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 0.</td></tr>
</table>

13.5.4.11 TBI Control Register (TBICON)

Figure 13-109 shows 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	TBIPA offset + 0x11															

Figure 13-109. TBI Control Register Definition

Table 13-109 describes the fields of the TBICON register.

Table 13-109. TBICON Field Descriptions

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. 0 Normal operation 1 Resets the functional modules in 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
12–15	—	Reserved

13.6 Functional Description

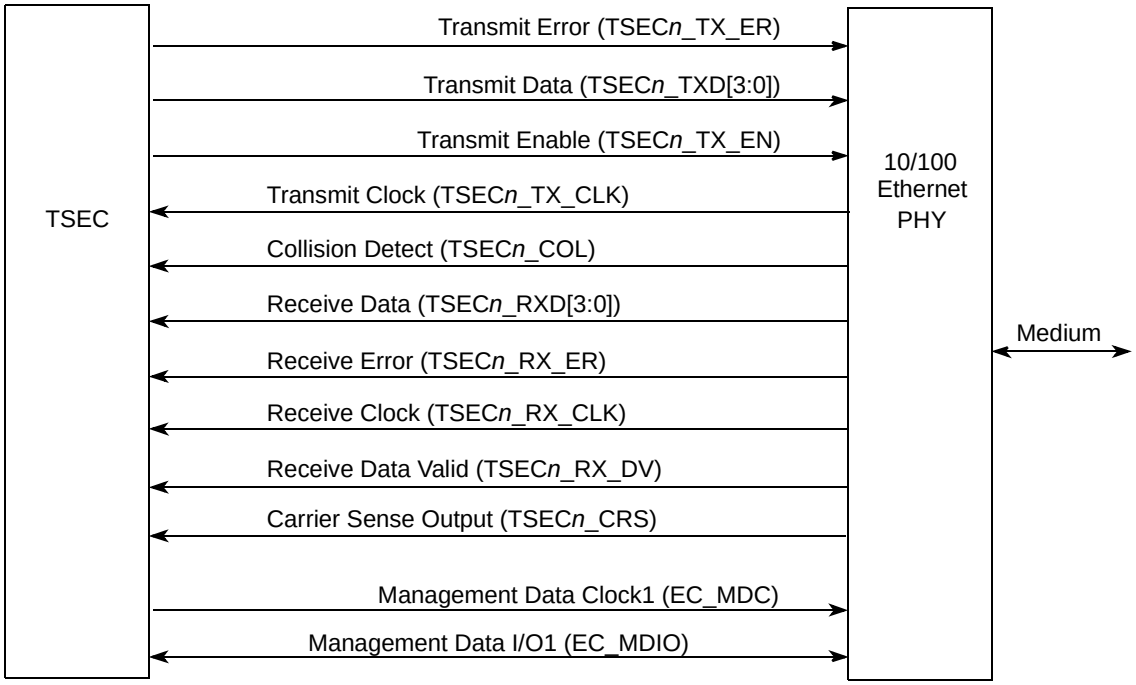
This section describes many of the functions of the TSEC controller.

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 each PHY follows the same convention. (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 13-110 shows the basic components of the MII including the signals required to establish TSEC module connection with a PHY.



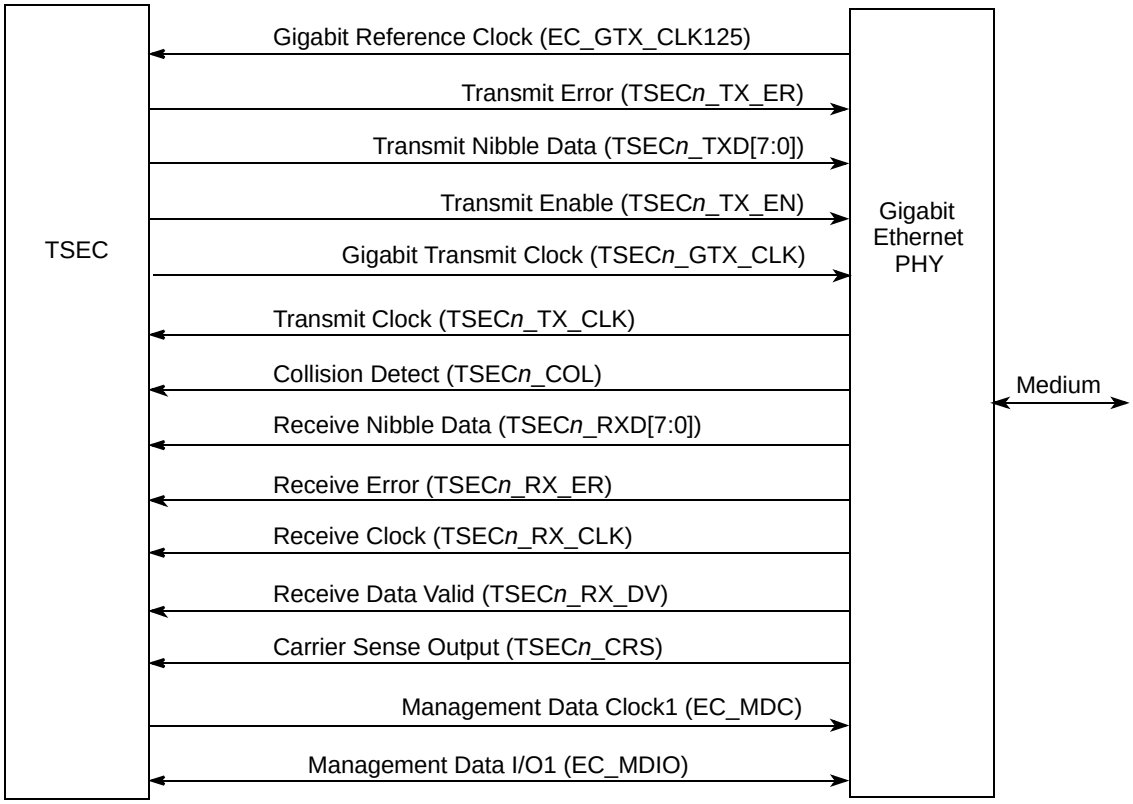
[1] The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 13-110. 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 13-111 shows the basic components of the GMII including the signals required to establish the TSEC module connection with a PHY.



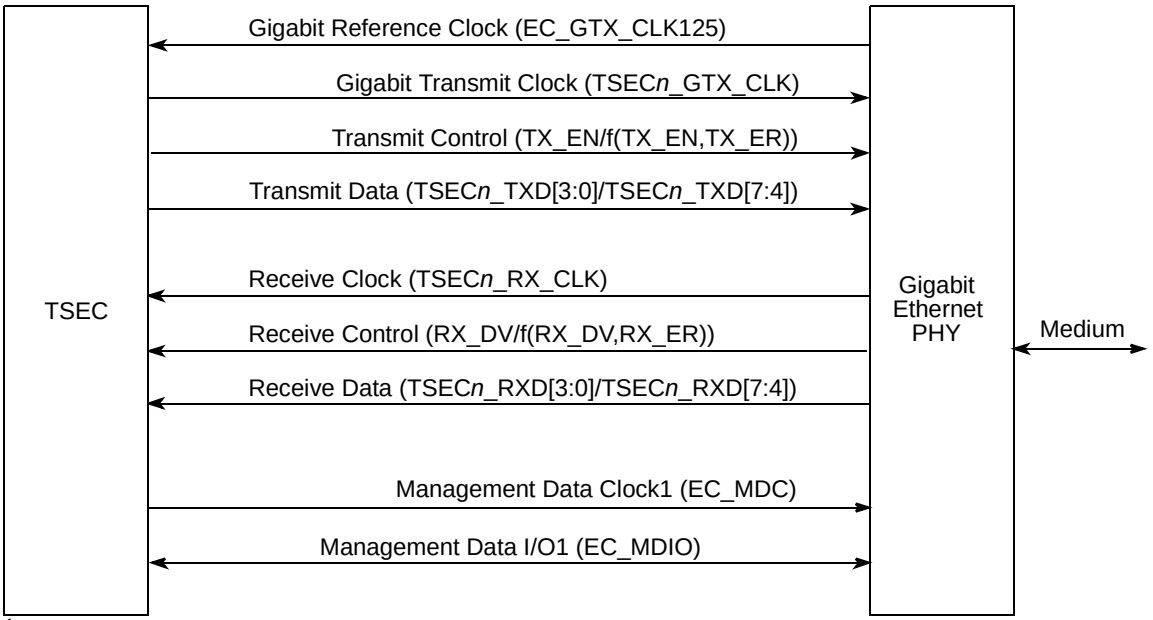
[1] The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 13-111. 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 signals required to interconnect the MAC and the PHY from a maximum of 28 signals (GMII) to 15 signals (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 125 MHz, and for 10/100 operation, the clocks operate at 2.5 MHz or 25 MHz, respectively. Figure 13-112 shows 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.

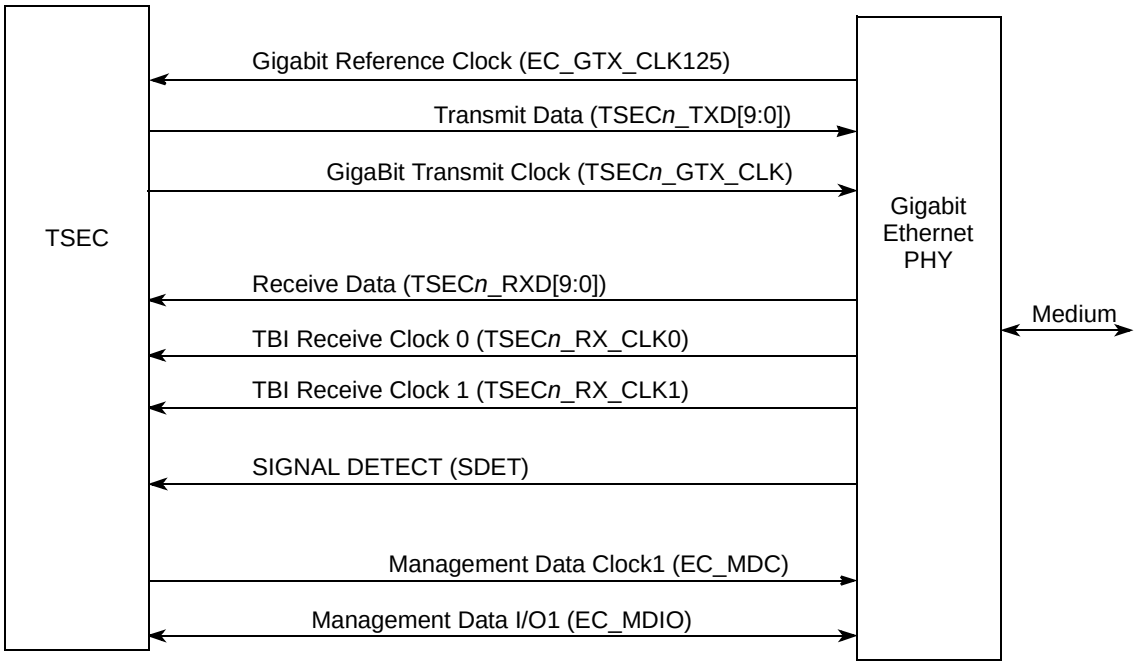


[1] The management signals (EC_MDC and EC_MDIO) are common to all of the gigabit Ethernet controllers module connections in the system, assuming that each PHY has a different management address.

Figure 13-112. 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 13-113 shows 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.



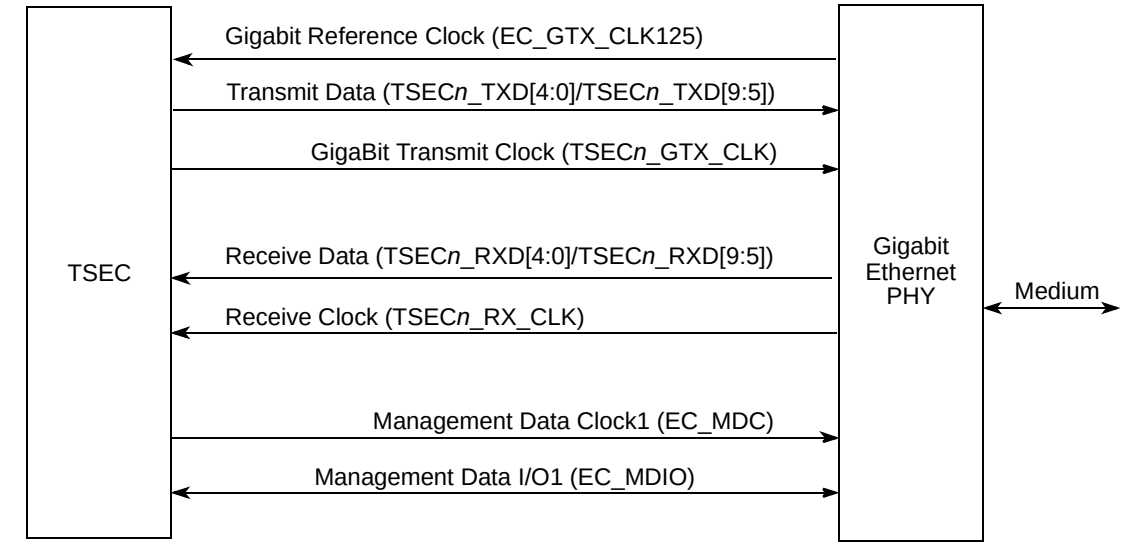
[1] The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 13-113. TSEC-TBI Connection

A TBI interface has 27 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 signal count version of a SerDes interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 13-114 shows 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 13-114. 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 13-110 describes the signal multiplexing for GMII, MII and TBI interfaces.

Table 13-110. GMII, MII, and TBI Signal Multiplexing

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			GMII Interface Frequency [MHz] 125 Voltage[V] 3.3			MII Interface Frequency [MHz] 25 Voltage[V] 3.3			TBI Interface Frequency [MHz] 62.5 Voltage[V] 3.3		
Signals (TSEC*n*_)	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1				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
TxD[6]	O	1	TxD[6]	O	1				TCG[6]	O	1
TxD[7]	O	1	TxD[7]	O	1				TCG[7]	O	1

Table 13-110. GMII, MII, and TBI Signal Multiplexing (continued)

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			GMII Interface Frequency [MHz] 125 Voltage[V] 3.3			MII Interface Frequency [MHz] 25 Voltage[V] 3.3			TBI Interface Frequency [MHz] 62.5 Voltage[V] 3.3		
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 13-111 describes the signal multiplexing for RGMII and RTBI interfaces.

Table 13-111. RGMII and RTBI Signal Multiplexing

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			RGMII Interface Frequency [MHz] 125 Voltage[V] 2.5			RTBI Interface Frequency [MHz] 62.5 Voltage[V] 2.5		
Signals (TSEC*n*_)	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1						
TxD[0]	O	1	TxD[0]/TxD[4]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1	TCG[1]/TCG[6]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1	TCG[3]/TCG[8]	O	1

Table 13-111. RGMII and RTBI Signal Multiplexing (continued)

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			RGMII Interface Frequency [MHz] 125 Voltage[V] 2.5			RTBI Interface Frequency [MHz] 62.5 Voltage[V] 2.5		
Signals (TSEC*n*_)	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**	**Signals (TSEC*n*_)**	**I/O**	**No. of Signals**
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						
Sum		25	**Sum**		12	**Sum**		12

Table 13-112 describes the signals shared by all interfaces.

Table 13-112. Shared Signals

Signals	I/O	# of signals	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 section 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 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. The module memory map in Table 13-3 lists all the TSEC registers. Table 13-113 describes the minimum steps for register initialization.

Table 13-113. Steps of Minimum Register Initialization

Description
1. Set, then clear MACCFG1 [Soft_Reset]
2. Initialize MACCFG2
3. Initialize MAC station address

Table 13-113. Steps of Minimum Register Initialization (continued)

Description
4. Set up the PHY using the MII Mgmt Interface
5. Configure the TBI control to TBI or GMII
6. Clear IEVENT
7. Initialize IMASK
8. Initialize IADDR*n*
9. Initialize GADDR*n*
10. Initialize RCTRL
11. Initialize DMACTRL

After the registers are initialized, the user must execute the following steps in the order described below to bring the TSEC into a functional state (out of reset):

1. For the transmission of Ethernet frames, TxBDs must first be built in memory, linked together as a ring, and pointed to by the TBASE register. A minimum of two buffer descriptors per ring is required. Setting the ring to a size of one causes the same frame to be transmitted twice.
2. Likewise, for the reception of Ethernet frames, the receive queue must be ready, with its RxBD pointed to by the RBASE register. Both transmit and receive can be gracefully stopped after transmission and reception begins.
3. Write to MACCFG1 and set the appropriate bits. These need to include Rx_EN and Tx_EN. To enable flow control, Rx_Flow and Tx_Flow must also be set.
4. Clearing DMACTRL[GTS] triggers the transmission of frame data if the transmitter had been previously stopped. DMACTRL[GRS] must be cleared if the receiver had been previously stopped. See Section 13.5.3.1.7, “DMA Control Register (DMACTRL),” and Section 13.6.3.1, “Transmit Data Buffer Descriptor (TxBD),” 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 are used, the user must load the pointers into the TBASE register. Likewise if a new set of Rx buffer descriptors are used, the RBASE register must be written with the new pointer.

Following is a procedure to gracefully reset and reconfigure the MAC:

1. Set GTS bit in DMACTRL register.
2. Poll GTSC bit in IEVENT register until detected as set.
3. Clear both Rx_EN and Tx_EN bits in MACCFG1.
4. Wait for a period of 9.6 Kbytes worth of data on the interface (~8ms worst case).
5. Set GRS bit in DMACTRL register.
6. Poll GRSC bit in IEVENT register until detected as set.
7. Set Soft_Reset bit in MACCFG1 register.
8. Clear Soft_Reset bit in MACCFG1 register.
9. Load TBASE with new TxBD pointer.
10. Load RBASE with new RxBD pointer.
11. Set up other MAC registers (MACCFG2, MAXFRM, and so on).
12. Set WWR and WOP bits in DMACTRL register.
13. Clear THLT bit in TSTAT register and QHLT bit in RSTAT register by writing 1 to these bits.
14. Clear GRS/GTS bits in DMACTRL. (Do not change other bits.)
15. Enable Tx_EN/Rx_EN in MACCFG1 register.

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, a transmit-on-demand function may be emulated while in polling mode by using the graceful-transmit-stop feature. First, clear the IMASK[GTSCEN] bit to mask the graceful-transmit-stop complete interrupt. Next set, then immediately clear the DMACTRL[GTS] bit. Clear the resulting IEVENT[GTSC] bit. Finally, the IMASK[GTSCEN] bit may be set once again.

There is one internal buffer for out-of-sequence flow control frames. While the 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. The pause duration is defined by the received pause control frame and begins when the frame was first received. In addition, the transmitter supports transmission of flow control frames via TCTRL[TFC_PAUSE]. The transmit pause frame is generated internally based on the PAUSE register that defines the pause value to be sent. Note that it is possible to send a pause frame while the pause timer has not expired.

The MAC automatically appends FCS (32-bit CRC) bytes to the frame if any of the following values are set:

- TxBD[PAD/CRC] is set in first TxBD
- TxBD[TC] is set in first TxBD
- MACCFG2[PAD/CRC] is set
- MACCFG2[CRC EN] is set

Following the transmission of the FCS, the Ethernet controller writes the frame status bits into the BD and clears TxBD[R]. If the end of the current buffer is reached and TxBD[L] is cleared (a frame is comprised of multiple buffer descriptors), only TxBD[R] is cleared.

For both half- and full-duplex modes, an interrupt can be issued depending on TxBD[I]. The Ethernet controller then proceeds to the next TxBD in the table. In this way, the core can be interrupted after each frame, after each buffer, or after a specific buffer is sent. If TxBD[PAD/CRC] is set, the Ethernet controller pads any frame shorter than 64 bytes.

To pause transmission or rearrange the transmit queue, set DMACTRL[GTS]. This can help in transmitting expedited data ahead of previously linked buffers or for error situations. If GTS is set, the 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 GTS is cleared, the TSEC resumes transmission with the next frame.

While the TSEC is in 10/100 Mbps mode it sends bytes least-significant nibble first and each nibble is sent lsb first. While it is in 1-Gbps 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 address recognition, CRC checking, short frame checking, and maximum frame-length checking. The receiver can also force frame headers and buffer descriptors to be allocated into the L2 cache. See Section 13.6.4, “Data Extraction to the L2 Cache,” for additional information.

After a hardware reset, the software driver clears the RSTAT register and sets MACCFG1[RX_EN]. The Ethernet receiver is enabled and immediately starts processing receive frames. If TSEC*n*_RX_DV is asserted and TSEC*n*_COL remains negated, the MAC strips a valid preamble/SFD (start of frame delimiter) header and begins processing the frame. If a valid header is not found, the frame is ignored.

If the receiver detects the first bytes of a frame, the TSEC controller begins to perform the frame recognition function through destination address (DA) recognition (See Section 13.6.2.6, “Frame Recognition,” for additional information.). 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), the 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 RxBDs are used; thus, no collision frames are presented to the user except late collisions, which indicate LAN problems.

The RxBD length is determined by the MRBL field in the maximum receive buffer length register (MRBL). The smallest valid value is 64 bytes. During reception, the Ethernet controller checks for frames that are too short or too long. After the frame ends (CRS is negated), the receive CRC field is checked and written to the data buffer. The data length written to the last RxBD in the Ethernet frame is the length of the entire frame, which enables the software to recognize a frame-too-long condition.

Receive frames are not truncated if they exceed maximum frame bytes in the MAC's maximum frame register if MACCFG2[Huge Frame] is set, yet the babbling receiver error interrupt occurs (IEVENT[BABR] is set) and RxBD[LG] is set.

After the receive frame is complete, the Ethernet controller sets RxBD[L], updates the frame status bits in the RxBD, and clears RxBD[E]. If RxBD[I] is set, the Ethernet controller next generates an interrupt (that can be masked) indicating that a frame was received and is in memory. The Ethernet controller then waits for a new frame.

To interrupt reception or rearrange the receive queue, DMACTRL[GRS] must be set. If this bit is set, the 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

TSEC automatically gathers network statistics required for RMON without needing to receive all addresses. The 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 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 13-115 is a flowchart for address recognition on received frames that is used to explain the concept. In the actual implementation most of the decision points shown in the figure actually occur simultaneously.

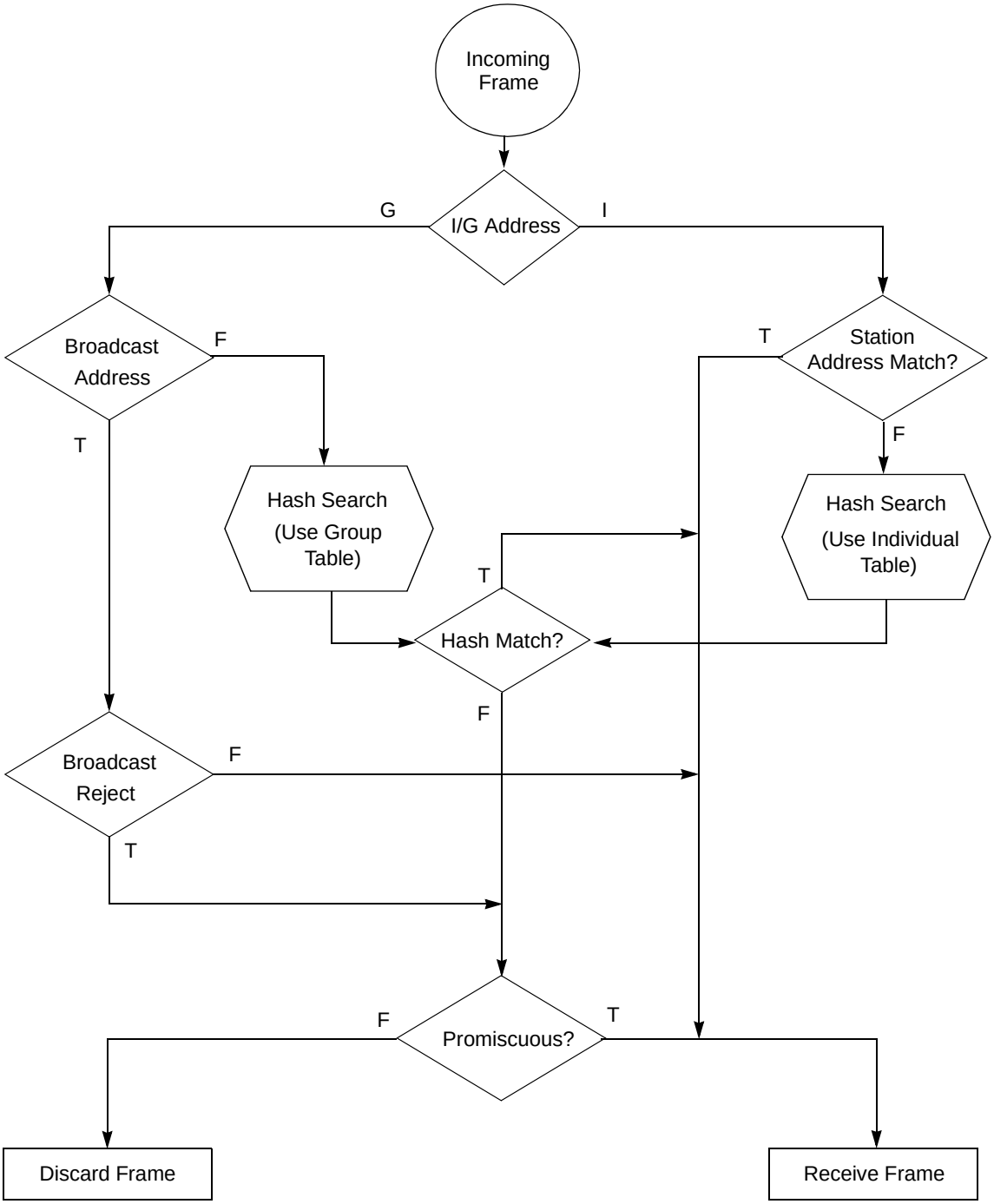


Figure 13-115. Ethernet Address Recognition Flowchart

The Ethernet controller compares the destination address field of the received frame with the physical address the user programs in the station address registers (MACSTNADDR1 and MACSTNADDR2). If the DA does not match the station address, then the controller performs address recognition on multiple individual addresses using the IADDR*n* hash table. The user must

write zeros to the hash in order to avoid a hash match, and ones to station address in order to avoid individual address match, or the user can turn on promiscuous mode. (See Section 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.

13.6.2.6.3 CRC Computation Examples

There are many algorithms for calculating the CRC value of a number. Refer to the RFC 3309 standard, which can be found at http://www.faqs.org/rfcs/rfc3309.html, to compute the CRC value for the purposes of TSEC. The RFC 3309 algorithm uses the following polynomial to calculate the CRC value: x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x1+x0 or 0x04c11db7.

Given a destination MAC address of DA=01000CCCCCCC, the algorithm results in a CRC remainder value of 0xA29F4BBC.

Bit-reversing the low-order byte of the CRC value (0xBC) yields BR_CRC = 0x3D = 0b00111101

The high-order 3-bits of the new BR_CRC value are used to select which 32-bit register (of the 8) to use. This example maps the DA to register 1.

High-order 3 bits of BR_CRC: HO_CRC = 0b001 = 1

The low-order 5 bits are used to select which bit to set in the given register (with a value of 0 setting 0x8000_0000 and 31 setting 0x0000_0001). Therefore, the example DA maps to bit 29 of register 1.

Low-order 5 bits of BR_CRC: LO_CRC = 0b11101 = 29

Therefore, GADDR1 is ORed with the value 0x0000_0004.

Additional calculated examples follow:

Example 1:

- Destination MAC address: DA = 01005E000128
- CRC remainder value: CRC = 0x821D6CD3
- Bit-reversed least-significant byte of CRC value: BR_CRC = 0xCB = 0b11001011
- High-order 3 bits of BR_CRC: HO_CRC = 0b110 = 6
- Low-order 5 bits of BR_CRC: LO_CRC = 0b01011 = 11
- GADDR6 = 0x0010_0000

Example 2:

- Destination MAC address: DA = 0004F0604F10
- CRC remainder value: CRC = 0x1F5A66B5
- Bit-reversed least-significant byte of CRC value: BR_CRC = 0xAD = 0b10101101
- High-order 3 bits of BR_CRC: HO_CRC = 0b101 = 5
- Low-order 5 bits of BR_CRC: LO_CRC = 0b01101 = 13
- GADDR5 = 0x0004_0000

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 13-114 lists the flow-control frame structure.

Table 13-114. Flow Control Frame Structure

Size [Octets]	Description	Value	Comment
7	Preamble		
1	SFD		Start frame delimiter
6	Destination address	01-80C2-00-00-01	Multicast address reserved for use in MAC frames
6	Source address		
2	Length/type	88-08	Control frame type
2	MAC opcode	00-01	Pause command
2	MAC parameter		Pause time as defined by the PTV[PT] field. The pause period is measured in pause_quanta, a speed-independent constant of 512 bit-times (unlike slot time). The most-significant octet is sent first.
2	Extended MAC parameter		Extended pause control parameter as defined by the PTV[PTE] field. The most-significant octet is sent first.
40	Reserved	—	
4	FCS		Frame check sequence (CRC)

If flow-control mode is enabled (MACCFG1[Rx_Flow] is set) and the receiver identifies a pause-flow control frame, transmission stops for the time specified in the control frame. During this pause, only a control frame can be sent (TCTRL[TFC_PAUSE] is set). Normal transmission resumes after the pause timer stops counting. If another pause-control frame is received during the pause, the period changes to the new value received.

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 TXF] were set. If the transmit speed is fast or the interrupt delay is long, more than one transmit buffer may have been sent by the 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 13-115.
- Obtain data from the RxBD if IEVENT[RXC,RXB or RXF] is set. If the receive speed is fast or the interrupt delay is long, the 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 13-116.

- Clear any set halt bits in TSTAT and RSTAT registers, or DMACTRL[GTS] and DMACTRL[GRS].
- Continue normal execution.

Table 13-115. 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 13-116. 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 lasting for the duration specified in the received pause control frame and beginning when the frame was first received.	None
RXB	Receive buffer: A receive buffer descriptor, that is not the last one of the frame, was updated.	Programmable “write with response” RxBD to memory before setting IEVENT[RXB].
RXF	Receive frame: A frame was received and the last receive buffer descriptor (RxBD) of that frame was updated.	Programmable “write with response” to memory on the last RxBD before setting IEVENT[RXF].

13.6.2.8.1 Interrupt Coalescing

Interrupt coalescing offers the user the ability to contour the behavior of the TSEC with regard to frame interrupts. Separate but identical mechanisms exist for handling both transmitted frames and received frames. While interrupt coalescing is enabled a transmit or receive frame interrupt (resulting from the interrupt bit (I) of the buffer descriptor in question and appropriately-enabled by the IMASK register) is raised either when a counter threshold-defined number of frames is received/transmitted or the timer threshold-defined period of time has lapsed, whichever occurs first.

13.6.2.8.2 Interrupt Coalescing By Frame Count Threshold

To avoid interrupt bandwidth congestion due to frequent, consecutive interrupts, the user may enable and configure interrupt coalescing to deliberately group frame interrupts, reducing the total number of interrupts raised. The number of frames received or transmitted prior to an interrupt being raised is determined by the frame threshold field (ICFCT) in the appropriate interrupt coalescing configuration register (RXIC or TXIC). The frame threshold field may be assigned a value between 1 and 255. A value of 0 results in boundedly undefined behavior. Note that a value of 1 functionally defeats the advantages of interrupt coalescing since the frame threshold is reached with each frame received or transmitted. Once the number of frames transmitted or received reaches the threshold limit, an interrupt is raised (if the interrupt bit of the frame buffer descriptor(s) is set and if appropriately-enabled in the IMASK register), the threshold counter is reset, and then continues counting frames while the interrupt is active. The threshold counter is also reset if an interrupt is raised subject to the corresponding threshold timer.

13.6.2.8.3 Interrupt Coalescing By Timer Threshold

To avoid stale frame interrupts, the user may also assign a timer threshold, beyond which any frame interrupts not yet raised are forced (if the interrupt bit of the frame buffer descriptor(s) is set and if enabled in the IMASK register). The timer threshold fields of the receive and transmit interrupt coalescing configuration registers (RXIC[ICTT] and TXIC[ICTT]) are defined in units equivalent to 64 TSEC interface clocks. That is, one timer threshold unit is 26.5 µs, 2.56 µs, or 512 ns, corresponding to interface modes 10 Mbps, 100 Mbps, or 1 Gbps, respectively.

After transmitting a frame (with TXBD[I] set, causing IEVENT[TXF] to be set), the transmit interrupt coalescing threshold timer begins counting. When the timer threshold has been reached an interrupt is raised if the interrupt bit of the buffer descriptor is set and IMASK[TXFEN] is set.

After receiving a frame (with RXBD[I] set, causing IEVENT[RXF] to be set), the receive interrupt coalescing threshold timer begins counting. When the timer threshold has been reached an interrupt is raised if the interrupt bit of the buffer descriptor is set and IMASK[RXFEN] is set.

The interrupt coalescing timer thresholds (transmit and receive, operating independently) may be values ranging from 1 to 65535. A value of 0 results in boundedly undefined behavior. Table 13-117 specifies the range of possible timing thresholds subject to the interface frequency and the value of RXIC[ICTT] or TXIC[ICTT].

Table 13-117. Interrupt Coalescing Timing Threshold Ranges

TSEC Interface Format and Frequency	Interrupt Coalescing Threshold Time	
	Minimum (ICTT=1)	Maximum (ICTT=65535)
10Base-T at 2.5 MHz	25.6 µs	1.678 s
100Base-T at 25 MHz	2.56 µs	167.8 ms
1000Base-T at 125 MHz	512 ns	33.55 ms

The transmit or receive timer threshold counter is reset when a corresponding interrupt is raised and begins counting again upon deassertion of that interrupt and once IEVENT[TXF or RXF] is set.

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.

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.

Programming note: When the TSEC encounters a halt condition (TSTAT[THLT] is set), it stops processing the frame at the current TxBD. The TSEC relies on the user to manage the buffer descriptor pointer, TBPTR, or the buffer descriptor queue before resuming transmissions. Once the TSEC resumes, it fetches the TxBD pointed to by TBPTR.

Transmission errors are described in Table 13-118.

Table 13-118. Transmission Errors

Error	Response
Transmitter underrun	The controller sends 32 bits that ensure a CRC error, terminates buffer transmission, sets TxBD[UN], closes the buffer, sets IEVENT[XFUN] and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Retransmission attempts limit expired	The controller terminates buffer transmission, sets TxBD[RL], closes the buffer, sets IEVENT[CRL/XDA] and IEVENT[TXE]. Transmission resumes after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Excessive defer abort	The controller terminates buffer transmission, sets TxBD[DEF], closes the buffer, sets IEVENT[CRC/XDA], and IEVENT[TXE]. Transmission resumes after TSTAT[THLT] is cleared.
Late collision	The controller terminates buffer transmission, sets TxBD[LC], closes the buffer, sets IEVENT[LC] and IEVENT[TXE]. The controller resumes transmission after TSTAT[THLT] is cleared (and DMACTRL[GTS] is cleared).
Memory Read Error	A system bus error occurred during a DMA transaction. The controller sets IEVENT[EBERR], DMA stops sending data to the FIFO which causes an underrun error but IEVENT[XFUN] is not set. The TSTAT[THLT] is set. Transmits are continued once TSTAT[THLT] is cleared.
Babbling Transmit Error	A frame is transmitted which exceeds the MAC's Maximum Frame Length and MACCFG2[Huge Frame] is a 0. The controller sets IEVENT[BABT] and continues without interruption. TxBD[TXTRUNC] is set in the last TxBD (TxBD[L] is set) of the frame.

Reception errors are described in Table 13-119.

Table 13-119. Reception Errors

Error	Description
Overrun error	The Ethernet controller maintains an internal FIFO buffer for receiving data. If a receiver FIFO buffer overrun occurs, the controller sets RxBD[OV], sets RxBD[L], closes the buffer, and sets IEVENT[RXF], The receiver then enters hunt mode (seeking start of a new frame).
Busy error	A frame is received and discarded due to a lack of buffers. The controller sets IEVENT[BSY]. In addition, the RSTAT[QHLT] bit is set. The halted queue resumes reception once the user clears the RSTAT[QHLT] bit.
Non-octet error (dribbling bits)	The Ethernet controller handles a nibble of dribbling bits if the receive frame terminates as non-octet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, the frame non-octet aligned (RxBD[NO]) error is reported, IEVENT[RXF] is set, and the alignment error counter increments. The 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[RXF]. 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[QHLT] bit is set. The halted queue resumes reception once the RSTAT[QHLT] bit is cleared.
Babbling Receive Error	A frame is received that exceeds the MAC's maximum frame length. The controller sets IEVENT[BABR] and continues

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 8-byte data BD format is similar to the MPC8260 BD model.

Data buffers are used in the transmission and reception of Ethernet frames (see Figure 13-116). 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 in the buffer descriptor are used to communicate status/control information between the TSEC and the software driver.

The status field of the BD is 16-bit field, as is the length field. The data buffer pointer is a 32-bit field. Therefore, the BDs should be accessed with the following C structure:

```
typedef unsigned short uint_16; /* choose 16-bit native type */

typedef unsigned int uint_32; /* choose 32-bit native type */

typedef struct bd_struct {

  uint_16 flags;

  uint_16 length;

  uint_32 bufptr;

};
```

Because there is no next BD pointer in the transmit/receive BD (see Figure 13-117), 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 TBASE and RBASE that point to the beginning transmit and receive BDs for the TSEC.

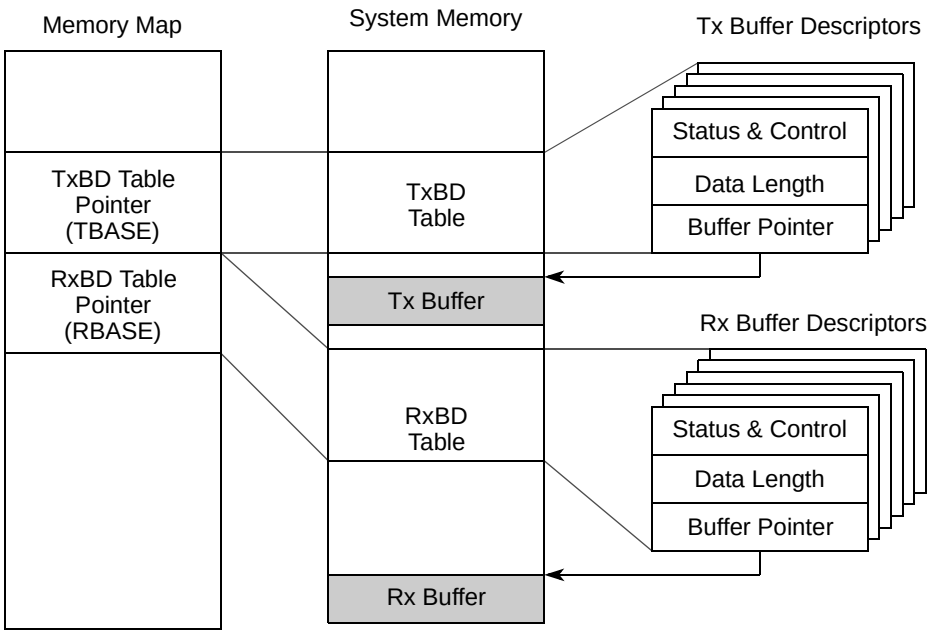


Figure 13-116. Example of TSEC Memory Structure for BD

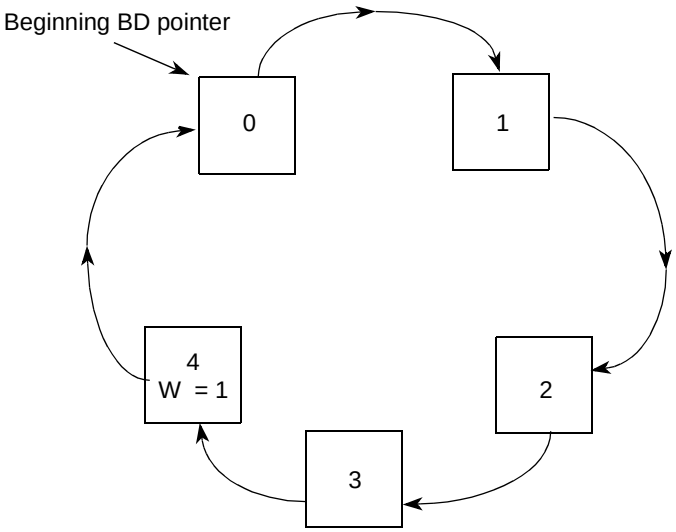


Figure 13-117. Buffer Descriptor Ring

13.6.3.1 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/CRC, 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 13-118 shows the TxBD.

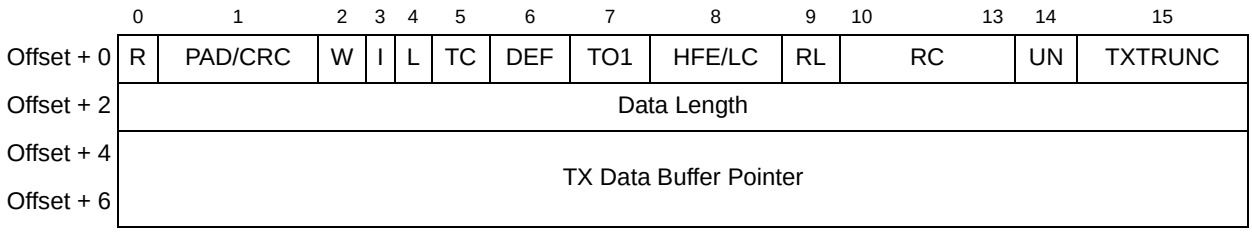


Figure 13-118. Transmit Buffer Descriptor

The TxBD fields are detailed in Table 13-120.

Table 13-120. 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.
Offset + 0	1	PAD/ CRC	PAD/CRC. Padding and CRC attachment for frames. (Valid only while it is set in the first BD and MACCFG2[PAD/CRC] is cleared.) If MACCFG2[PAD/CRC] is set, this bit is ignored. 0 Do not add padding to short frames. No CRC is appended unless TxBD[TC] is set. 1 Add PAD/CRCs to frames. PAD bytes are inserted until the length of the transmitted frame equals 64 bytes. 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] is cleared and MACCFG2[CRC EN] is cleared.) If MACCFG2[PAD/CRC] is set or MACCFG2[CRC EN] is set, this bit is ignored. 0 End transmission immediately after the last data byte with no hardware generated CRC appended, unless TxBD[PAD/CRC] is set. 1 Transmit the CRC sequence after the last data byte.

Table 13-120. Transmit Data Buffer Descriptor (TxBD) Field Descriptions

Offset	Bits	Name	Description
Offset + 0	6	DEF	Defer indication. Hardware updates this bit if an excessive defer condition occurs. 0 This frame was not deferred. 1 If HAFDUP[EXCESS_DEFER]=1, this frame did not have a collision before it was sent but it was sent late because of deferring. If HAFDUP[EXCESS_DEFER]=0, this frame was aborted and not sent.
Offset + 0	7	TO1	Transmit software ownership. This read/write bit may be utilized by software, as necessary. Its state does not affect the hardware nor is it affected by the hardware.
Offset + 0	8	HFE/LC	Huge Frame Enable (written by user)/Late collision (written by TSEC) Valid only while it is set in first BD and the MACCFG2[Huge Frame] is cleared. If MACCFG2[Huge Frame] is set, this bit is ignored. 0 Truncate transmit frame if its length is greater than the MAC's Maximum Frame Length register. 1 Do not truncate the transmit frame. Late collision. Written by TSEC 0 No late collision 1 A collision occurred after 64 bytes are sent. The TSEC terminates the transmission and updates LC.
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 0000 The frame is sent correctly the first time or if RL is set then the retry limit has been reached *nnnn* One or more attempts were needed to send the transmit frame. If this field is 15, then 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
Offset + 0	14	UN	Underrun. Written by 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	TXTRUNC	TX truncation. Set in the last TxBD (TxBD[L] is set) when IEVENT[BABT] occurs for the frame
Offset + 2	0–15	Data Length	Data length is the number of octets the TSEC transmits 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.

13.6.3.2 Receive Buffer Descriptor (RxBD)

In the RxBD the user initializes the E, I, and W bits in the first word and the pointer in second word. If the data buffer is used, the 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 the TSEC if the L bit is set. The first word of the RxBD contains control and status bits. Its format is detailed in Figure 13-119 below.

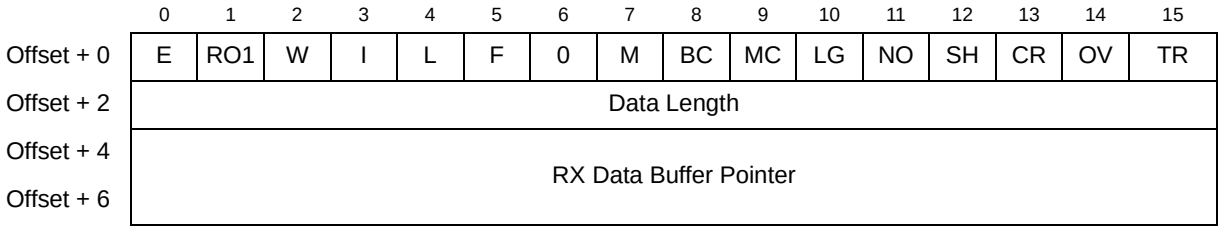


Figure 13-119. Receive Buffer Descriptor

Table 13-121 describes the fields of the RxBD.

Table 13-121. Receive Buffer Descriptor Field Descriptions

Offset	Bits	Name	Description
Offset + 0	0	E	Empty. Written by TSEC (when cleared) and by user (when set). 0 The data buffer associated with this BD is filled with received data, or data reception is aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	1	RO1	Receive software ownership bit. Reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	2	W	Wrap. Written by user 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in RBASE.
Offset + 0	3	I	Interrupt. Written by user 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[RXB] or IEVENT[RXF] are set after this buffer is serviced. This bit can cause an interrupt if enabled (IMASK[RXBEN] is set or IMASK[RXFEN] is set). If the user wants to be interrupted only if RXF occurs, then the user must disable RXB (IMASK[RXBEN] is cleared) and enable RXF (IMASK[RXFEN] is set).
Offset + 0	4	L	Last in frame. Written by 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 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 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.

Table 13-121. Receive Buffer Descriptor Field Descriptions (continued)

Offset	Bits	Name	Description
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 TSEC. (Only valid if L is set.) A frame length greater than maximum frame length was recognized while MACCFG2[Huge Frame] was set. Note, if MACCFG2[Huge Frame] is cleared, the frame is truncated to the value programmed in the Maximum Frame Length register.
Offset + 0	11	NO	Rx non-octet aligned frame. Written by TSEC. (Only valid if L is set.) A frame that contained a number of bits not divisible by eight was received.
Offset + 0	12	SH	Short frame. Written by 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 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 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 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] 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 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 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.

13.6.4 Data Extraction to the L2 Cache

Some applications require the ability to identify selected portions of data within a frame's data payload. This process is called extraction; although, the data is not truly removed. Rather than literarily extracting a section of the data and copying it into a new memory location, the data is placed in the L2 cache. This allows the processor to quickly access critical frame information as soon as the processor is ready without having to first fetch the data from main memory. This reduction in latency can result in a substantial improvement in packet processing throughput.

Extraction functionality is controlled and configured with ATTR and ATTRELI. See Section 13.5.3.9.1, "Attribute Register (ATTR)," and Section 13.5.3.9.2, "Attribute Extract Length and Extract Index Register (ATTRELI)," for specific register information.

13.7 Initialization/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 13-122 describes the signal configurations required for MII interface mode.

Table 13-122. MII Interface Mode Signal Configuration

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			MII Interface Frequency [MHz] 25 Voltage[V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	Leave Unconnected	O	
TX_CLK	I	1	TX_CLK	I	1
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	TxD[2]	O	1
TxD[3]	O	1	TxD[3]	O	1
TxD[4]	O	1	Leave Unconnected	O	
TxD[5]	O	1	Leave Unconnected	O	
TxD[6]	O	1	Leave Unconnected	O	
TxD[7]	O	1	Leave Unconnected	O	
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1
RxD[3]	I	1	RxD[3]	I	1
RxD[4]	I	1	not used	I	
RxD[5]	I	1	not used	I	

Table 13-122. MII Interface Mode Signal Configuration (continued)

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			MII Interface Frequency [MHz] 25 Voltage[V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
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 13-123 describes the shared signals of the MII interface.

Table 13-123. Shared MII Signals

TSEC Signals	I/O	No. of signals	MII Signals	I/O	No. of signals	Function
EC_MDIO	I/O	1	EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	EC_MDC	O	1	Management interface Clock
EC_GTX_CLK125	I	1	not used	I	0	Reference Clock
Sum		3	Sum		2	

Table 13-124 describes the register initializations required to reconfigure the PHY to MII mode following initial auto-negotiation.

Table 13-124. MII Mode Register Initialization Steps

Have the TSEC*n*_GTX_CLK configuration signal pulled low for G/MII mode.
Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, for MII, half duplex operation. Set I/F Mode bit, MACCFG2[0000_0000_0000_0000_0111_0001_0000_0100] (This example has Full Duplex = 0, Preamble count = 7, PAD/CRC append = 1)

Table 13-124. MII Mode Register Initialization Steps (continued)

Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] Set station address to 02_60_8C_87_65_43, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] Set station address to 02_60_8C_87_65_43, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Assign a Physical address to the TBI so as to not conflict with the external PHY Physical address, TBIPA[0000_0000_0000_0000_0000_0000_0000_0101] Set to 05, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Reset the management interface. MIIMCFG[1000_0000_0000_0000_0000_0000_0000_0000]
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_000_0000_0000_0101] set source clock divide by 14, for example, to insure that EC_MDC clock speed is approximately 2.5 MHz.
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the TSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to the external PHY Auxiliary Control and Status Register to configure the PHY through the Management interface (overrides configuration signals of the PHY). MIIMADD[0000_0000_0000_0000_0000_0000_0001_1100]
Perform an MII Mgmt write cycle to the external PHY Writing to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0100]
Check to see if MII Mgmt write is complete Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Extended PHY control register 1 to set up the interface mode selection. MIIMADD[0000_0000_0000_0000_0000_0000_0001_0111]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0000]
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Mode control register to set up the interface mode selection. MIIMADD[0000_0000_0000_0000_0000_0000_0000_0000]

Table 13-124. MII Mode Register Initialization Steps (continued)

Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_00uu_00uu_0u00_0000] where u is user defined based on desired configuration.
Check to see if MII Mgmt write is complete Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
If auto-negotiation was enabled in the PHY, check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0000_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x00.
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle]. Set MIIMCOM[Read Cycle]. (Uses the PHY address (0) and Register address (1) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10 (AN Done and Link is up) MIIMSTAT ---> [0000_0000_0000_0000_0000_000_0010_0100] Other information about the link is also returned.(Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Check auto-negotiation attributes in the PHY as necessary.
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK, (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IADDR*n,* (Optional) IADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n,* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL, (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL, (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data. Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
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 13-125 describes the signal configurations required for GMII interface mode.

Table 13-125. GMII Interface Mode Signal Configuration

TSEC SIGNALS Frequency [MHz] 125 Voltage[V] 3.3/2.5			GMII INTERFACE Frequency [MHz] 125 Voltage[V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1			
TxD[0]	O	1	TxD[0]	O	1
TxD[1]	O	1	TxD[1]	O	1
TxD[2]	O	1	TxD[2]	O	1
TxD[3]	O	1	TxD[3]	O	1
TxD[4]	O	1	TxD[4]	O	1
TxD[5]	O	1	TxD[5]	O	1
TxD[6]	O	1	TxD[6]	O	1
TxD[7]	O	1	TxD[7]	O	1
TX_EN	O	1	TX_EN	O	1
TX_ER	O	1	TX_ER	O	1
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]	I	1
RxD[1]	I	1	RxD[1]	I	1
RxD[2]	I	1	RxD[2]	I	1
RxD[3]	I	1	RxD[3]	I	1
RxD[4]	I	1	RxD[4]	I	1
RxD[5]	I	1	RxD[5]	I	1
RxD[6]	I	1	RxD[6]	I	1
RxD[7]	I	1	RxD[7]	I	1
RX_DV	I	1	RX_DV	I	1
RX_ER	I	1	RX_ER	I	1
COL	I	1			
CRS	I	1			
Sum		25	Sum		22

Table 13-126 describes the shared signals of the GMII interface.

Table 13-126. Shared GMII Signals

TSEC Signals	I/O	No. of Signals	GMII Signals	I/O	No. of Signals	Function
EC_MDIO	I/O	1	EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	EC_MDC	O	1	Management interface Clock
EC_GTX_CLK125	I	1	EC_GTX_CLK125	I	1	Reference Clock
Sum		3	Sum		3	

Table 13-127 describes the register initializations required to reconfigure the PHY to GMII mode following initial auto-negotiation.

Table 13-127. GMII Mode Register Initialization Steps

Have the TSEC*n*_GTX_CLK configuration signal pulled low for G/MII mode.
Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, for GMII, Full duplex operation. Set I/F Mode bit. MACCFG2[0000_0000_0000_0000_0111_0010_0000_0101] (This example has Full Duplex = 1, Preamble count = 7, PAD/CRC append = 1)
Initialize ECNTRL, ECNTRL[0000_0000_0000_0000_0001_0000_0000_0000] (This example has Statistics Enable = 1)
Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] Set station address to 02_60_8C_87_65_43, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] Set station address to 02_60_8C_87_65_43, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Assign a Physical address to the TBI so as to not conflict with the external PHY Physical address, TBIPA[0000_0000_0000_0000_0000_0000_0000_0101] Set to 05, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Reset the management interface, MIIMCFG[1000_0000_0000_0000_0000_0000_0000_0000]
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_0000_0000_0101] Set source clock divide by 14, for example, to insure that EC_MDC clock speed is approximately 2.5 MHz.

Table 13-127. GMII Mode Register Initialization Steps (continued)

Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the TSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to the external PHY Auxiliary Control and Status Register to configure the PHY through the Management interface (overrides configuration signals of the PHY), MIIMADD[0000_0000_0000_0000_0000_0000_0001_1100]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0100]
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed
Set up the MII Mgmt for a write cycle to the external PHY Extended PHY control register 1 to set up the interface mode selection MIIMADD[0000_0000_0000_0000_0000_0000_0001_0111]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_0000_0000_0000_0000]
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Mode control register to set up the interface mode selection, MIIMADD[0000_0000_0000_0000_0000_0000_0000_0000]
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY register, MIIMCON[0000_0000_0000_0000_000u_00u1_0100_0000] where u is user defined based on desired configuration.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
If auto-negotiation was enabled in the PHY, check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0000_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x00
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle]. Set MIIMCOM[Read Cycle]. (Uses the PHY address (0) and Register address (1) placed in MIIMADD register), When MIIMIND[BUSY]=0, Read the MIIMSTAT register and check bit 10 (AN Done and Link is up), MIIMSTAT ---> [0000_0000_0000_0000_0000_000_0010_0100] Other information about the link is also returned.(Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Check auto-negotiation attributes in the PHY as necessary.

Table 13-127. GMII Mode Register Initialization Steps (continued)

Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK, (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IADDR*n,* (Optional) IADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n,* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL, (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL, (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data. Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
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 13-128 describes the signal configurations required for TBI interface mode.

Table 13-128. TBI Interface Mode Signal Configuration

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			TBI Interface Frequency [MHz] 62.5 Voltage[V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1	RX_CLK1	I	1
TxD[0]	O	1	TCG[0]	O	1
TxD[1]	O	1	TCG[1]	O	1
TxD[2]	O	1	TCG[2]	O	1
TxD[3]	O	1	TCG[3]	O	1
TxD[4]	O	1	TCG[4]	O	1

Table 13-128. TBI Interface Mode Signal Configuration (continued)

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			TBI Interface Frequency [MHz] 62.5 Voltage[V] 3.3		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
TxD[5]	O	1	TCG[5]	O	1
TxD[6]	O	1	TCG[6]	O	1
TxD[7]	O	1	TCG[7]	O	1
TX_EN	O	1	TCG[8]	O	1
TX_ER	O	1	TCG[9]	O	1
RX_CLK	I	1	RX_CLK0	I	1
RxD[0]	I	1	RCG[0]	I	1
RxD[1]	I	1	RCG[1]	I	1
RxD[2]	I	1	RCG[2]	I	1
RxD[3]	I	1	RCG[3]	I	1
RxD[4]	I	1	RCG[4]	I	1
RxD[5]	I	1	RCG[5]	I	1
RxD[6]	I	1	RCG[6]	I	1
RxD[7]	I	1	RCG[7]	I	1
RX_DV	I	1	RCG[8]	I	1
RX_ER	I	1	RCG[9]	I	1
COL	I	1		I	
CRS	I	1	SDET	I	1
Sum		25	Sum		24

Table 13-129 describes the shared signals for the TBI interface.

Table 13-129. Shared TBI Signals

TSEC Signals	I/O	No. of Signals	TBI Signals	I/O	No. of Signals	Function
EC_MDIO	I/O	1	EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	EC_MDC	O	1	Management interface Clock
EC_GTX_CLK125	I	1	EC_GTX_CLK125	I	1	Reference Clock
Sum		3	Sum		3	

Table 13-130 describes the register initializations required to reconfigure the PHY to TBI mode following initial auto-negotiation.

Table 13-130. TBI Mode Register Initialization Steps

Have the TSEC*n*_GTX_CLK configuration signal pulled high and EC_MDC signal pulled high for TBI mode. (TSEC attempts to auto-negotiate after system reset.)
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] (This example has Statistics Enable = 1)
Initialize MAC Station Address MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Initialize MAC Station Address MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_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.
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the TSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a read cycle to TBI Control register (write the TBI address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The TBI Control register is at offset address 0x0 from TBIPA.
Perform an MII Mgmt read cycle to verify state of TBI Control Register (Optional) Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT and look for AN Enable and other bit information.
Set up the MII Mgmt for a write cycle to TBI's AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0100] The AN Advertisement register is at offset address 0x04 from the TBI's address. (in this case 0x10)

Table 13-130. TBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's AN Advertisement register, MIIMCON[0000_0000_0000_0000_0000_0001_1010_0000] This advertises to the Link Partner that the TBI supports PAUSE and Full Duplex mode and does not support Half Duplex mode.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to TBI's Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] the Control register is at offset address 0x00 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the TBI to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0001] The Phy Status control register is at address 0x1 and in this case the PHY Address is 0x10.
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (6) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Expansion register and check bits 13 and 14 (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]

Table 13-130. TBI Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (5) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 2 and 3 (Remote Fault) and bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_000x_x110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IADDR*n* (Optional) IADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
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 13-131 shows the signals configurations required for RGMII interface mode.

Table 13-131. RGMII Interface Mode Signal Configuration

TSEC SIGNALS Frequency [MHz] 125 Voltage[V] 3.3/2.5			RGMII INRTERFACE Frequency [MHz] 125 Voltage[V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1			
TxD[0]	O	1	TxD[0]/TxD[4]	O	1
TxD[1]	O	1	TxD[1]/TxD[5]	O	1
TxD[2]	O	1	TxD[2]/TxD[6]	O	1
TxD[3]	O	1	TxD[3]/TxD[7]	O	1
TxD[4]	O	1			
TxD[5]	O	1			
TxD[6]	O	1			
TxD[7]	O	1			
TX_EN	O	1	TX_CTL (TX_EN/TX_ERR)	O	1
TX_ER	O	1			
RX_CLK	I	1	RX_CLK	I	1
RxD[0]	I	1	RxD[0]/RxD[4]	I	1
RxD[1]	I	1	RxD[1]/RxD[5]	I	1
RxD[2]	I	1	RxD[2]/RxD[6]	I	1
RxD[3]	I	1	RxD[3]/RxD[7]	I	1
RxD[4]	I	1			
RxD[5]	I	1			
RxD[6]	I	1			
RxD[7]	I	1			
RX_DV	I	1	RX_CTL (RX_DV/RX_ERR)	I	1

Table 13-131. RGMII Interface Mode Signal Configuration (continued)

TSEC SIGNALS Frequency [MHz] 125 Voltage[V] 3.3/2.5			RGMII INRTERFACE Frequency [MHz] 125 Voltage[V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
RX_ER	I	1			
COL	I	1			
CRS	I	1			
Sum		25	Sum		12

Table 13-132 describes the shared signals for the RGMII interface.

Table 13-132. Shared RGMII Signals

TSEC Signals	I/O	No. of Signals	RGMII Signals	I/O	No. of Signals	Function
EC_MDIO	I/O	1	EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	EC_MDC	O	1	Management interface Clock
EC_GTX_CLK125	I	1	EC_GTX_CLK125	I	1	Reference Clock
Sum		3	Sum		3	

Table 13-133 describes the register initializations required to reconfigure the PHY to RGMII mode following initial auto-negotiation.

Table 13-133. RGMII Mode Register Initialization Steps

Have the TSEC*n*_GTX_CLK configuration signal pulled low and EC_MDC signal pulled low for RGMII mode. TBI Control register's Auto-negotiation Enable and Reset bits are ignored.
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] (This example has RGMII 10-Mbps mode, Statistics Enable = 1)

Table 13-133. RGMII Mode Register Initialization Steps (continued)

Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
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.
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the TSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a write cycle to external the PHY AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0100] The AN Advertisement register is at offset address 0x04 from the external PHY address. (in this case 0x11)
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY AN Advertisement register, MIIMCON[0000_0000_0000_0000_u0uu_uuuu_uuuu_uuuu] Where u must be selected by the user for proper system configuration.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to the external PHY Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0001_0000_0000] The Control register is at offset address 0x00 from the external PHY address. (in this case 0x11)
Perform an MII Mgmt write cycle to the external PHY. Write to MII Mgmt Control with 16-bit data intended for the external PHY Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the external PHY to restart Auto-Negotiations using the configuration set in the AN Advertisement register.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0000_0010_0000_0001] The PHY Status register is at address 0x1 and in this case the PHY Address is 0x2.

Table 13-133. RGMII Mode Register Initialization Steps (continued)

Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10. (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (6) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Expansion register and check bits 13 and 14. (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0001_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x11) and Register address (5) placed in MIIMADD register) When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_000x_x110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IADDR*n* (Optional) IADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]
Initialize (Empty) Transmit Descriptor ring and fill buffers with Data Initialize TBASE, TBASE[LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_LLLL_L000]
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 13-134 describes the signal configurations required for RTBI interface mode.

Table 13-134. RTBI Interlace Mode Signal Configuration

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			RTBI Interface Frequency [MHz] 62.5 Voltage[V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
GTX_CLK	O	1	GTX_CLK	O	1
TX_CLK	I	1			
TxD[0]	O	1	TCG[0]/TCG[5]	O	1
TxD[1]	O	1	TCG[1]/TCG[6]	O	1
TxD[2]	O	1	TCG[2]/TCG[7]	O	1
TxD[3]	O	1	TCG[3]/TCG[8]	O	1
TxD[4]	O	1			
TxD[5]	O	1			
TxD[6]	O	1			
TxD[7]	O	1			
TX_EN	O	1	TCG[4]/TCG[9]	O	1
TX_ER	O	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

Table 13-134. RTBI Interlace Mode Signal Configuration (continued)

TSEC Signals Frequency [MHz] 125 Voltage[V] 3.3/2.5			RTBI Interface Frequency [MHz] 62.5 Voltage[V] 2.5		
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
RX_ER	I	1			
COL	I	1			
CRS	I	1		I	
Sum		25	Sum		12

Table 13-135 describes the shared signals for the RTBI interface.

Table 13-135. Shared RTBI Signals

TSEC Signals	I/O	No. of Signals	RTBI Signals	I/O	No. of Signals	Function
EC_MDIO	I/O	1	EC_MDIO	I/O	1	Management interface I/O
EC_MDC	O	1	EC_MDC	O	1	Management interface Clock
EC_GTX_CLK125	I	1	EC_GTX_CLK125	I	1	Reference Clock
Sum		3	Sum		3	

Table 13-136 describes the register initializations required to reconfigure the PHY to RTBI mode following initial auto-negotiation.

Table 13-136. RTBI Mode Register Initialization Steps

Have the TSEC*n*_GTX_CLK configuration signal pulled high and the EC_MDC signal pulled low for RTBI mode. (TSEC attempts to auto-negotiate after system reset.)
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] (This example has Statistics Enable = 1)

Table 13-136. RTBI Mode Register Initialization Steps (continued)

Initialize MAC Station Address, MACSTNADDR2[0110_0000_0000_0010_0000_0000_0000_0000] to 02608C:876543, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Initialize MAC Station Address, MACSTNADDR1[0100_0011_0110_0101_1000_0111_1000_1100] to 02608C:876543, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Assign a Physical address to the TBI, TBIPA[0000_0000_0000_0000_0000_0000_0001_0000] set to 16, for example. (Note that PHY configuration register addresses are not necessarily consistent in all PHYs.)
Setup the MII Mgmt clock speed, MIIMCFG[0000_0000_0000_0000_0000_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.
Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the TSEC MII Mgmt bus is idle.
Set up the MII Mgmt for a read cycle to TBI Control register (write the TBI's address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The TBI Control register is at offset address 0x0 from TBIPA.
Perform an MII Mgmt read cycle to verify state of TBI Control Register(0ptional) Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the TBI address and Register address placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT and look for AN Enable and other bit information.
Set up the MII Mgmt for a write cycle to TBI's AN Advertisement register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0100] The AN Advertisement register is at offset address 0x04 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Write to MII Mgmt Control with 16-bit data intended for TBI's AN Advertisement register, MIIMCON[0000_0000_0000_0000_0000_0001_1010_0000] This advertises to the Link Partner that the TBI supports PAUSE and Full Duplex mode and does not support Half Duplex mode.
Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Set up the MII Mgmt for a write cycle to TBI's Control register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0000] The Control register is at offset address 0x00 from the TBI's address. (in this case 0x10)
Perform an MII Mgmt write cycle to TBI. Writing to MII Mgmt Control with 16-bit data intended for TBI's Control register, MIIMCON[0000_0000_0000_0000_0001_0010_0000_0000] This enables the TBI to restart Auto-Negotiations using the configuration set in the AN Advertisement register.

Table 13-136. RTBI Mode Register Initialization Steps (continued)

Check to see if MII Mgmt write is complete. Read MII Mgmt Indicator register and check for Busy = 0, MIIMIND ---> [0000_0000_0000_0000_0000_0000_0000_0000] This indicates that the write cycle was completed.
Check to see if PHY has completed Auto-Negotiation. Set up the MII Mgmt for a read cycle to the PHY MII Mgmt register (write the PHY address and Register address), MIIMADD[0000_0000_0000_0000_0001_0000_0000_0001] The Phy Status control register is at address 0x1 and in this case the PHY Address is 0x10.
Perform an MII Mgmt read cycle of Status Register. Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (2) and Register address (2) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MIIMSTAT register and check bit 10 (AN Done) MIIMSTAT ---> [0000_0000_0000_0000_0000_0000_0010_0000] Other information about the link is also returned. (Extend Status, No pre, Remote Fault, An Ability, Link status, extend Ability)
Perform an MII Mgmt read cycle of AN Expansion Register. Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0110] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (6) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Expansion register and check bits 13 and 14. (NP Able and Page Rx'd) MII Mgmt AN Expansion ---> [0000_0000_0000_0000_0000_0000_0000_0110]
Perform an MII Mgmt read cycle of AN Link Partner Base Page Ability Register. (Optional) Setup MIIMADD[0000_0000_0000_0000_0001_0000_0000_0101] Clear MIIMCOM[Read Cycle] Set MIIMCOM[Read Cycle] (Uses the PHY address (0x10) and Register address (5) placed in MIIMADD register), When MIIMIND[BUSY]=0, read the MII Mgmt AN Link Partner Base Page Ability register and check bits 9 and 10. (Half and Full Duplex) MII Mgmt AN Link Partner Base Page Ability ---> [0000_0000_0000_0000_0000_000x_x110_0000]
Clear IEVENT register, IEVENT[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IMASK (Optional) IMASK[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize IADDR*n* (Optional) IADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize GADDR*n* (Optional) GADDR*n*[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize RCTRL (Optional) RCTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize DMACTRL (Optional) DMACTRL[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]

Table 13-136. RTBI Mode Register Initialization Steps (continued)

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

This chapter describes the DMA controller of this device.

14.1 Introduction

The DMA controller transfers blocks of data between the RapidIO controller, PCI, the local bus controller (LBC) interface, and the local address space, independent of the e500 core or external hosts.

14.1.1 Block Diagram

Figure 14-1 shows the block diagram of the DMA controller.

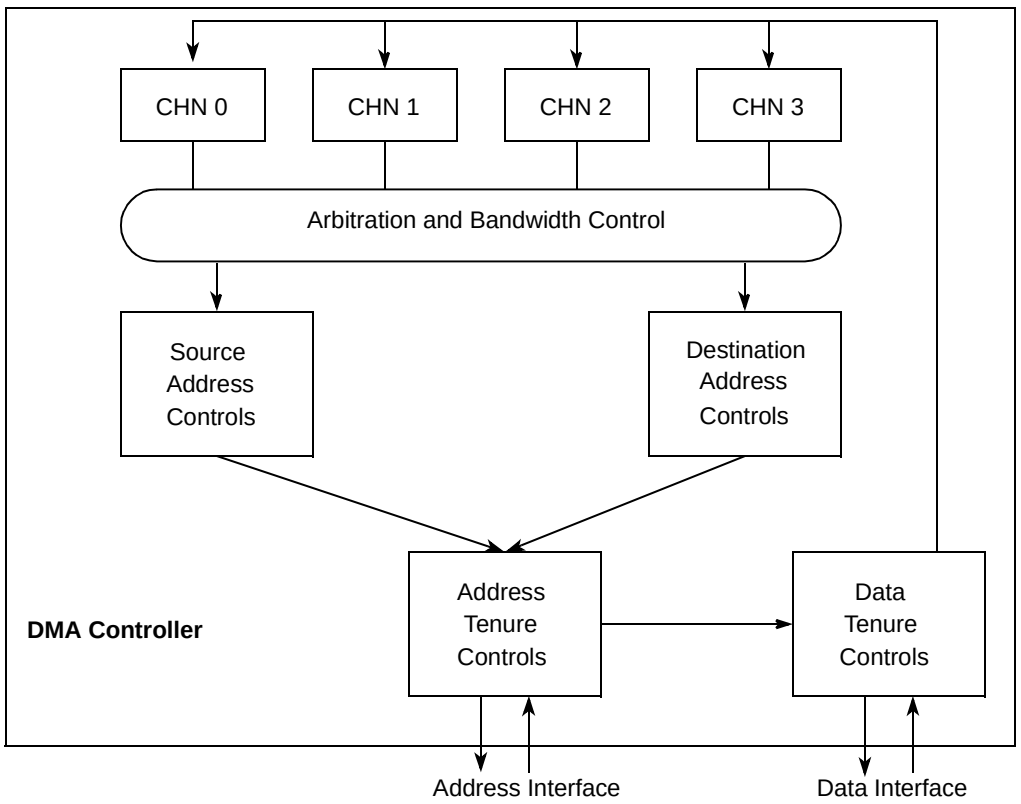


Figure 14-1. DMA Block Diagram

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 is a high-level block diagram of the DMA controller. Operations such as descriptor fetches and block transfers are initiated by each channel. A channel is selected by the arbitration logic and information is passed to the source and destination control blocks for processing. The source and destination blocks generate read and write requests to the address tenure engine, which manages the DMA master port address interface. After a transaction is accepted by the master port, control is transferred to the data tenure engine that manages the read and write data transfers. A channel remains active in the shared resources for the duration of the data transfer unless the allotted bandwidth per channel is reached.

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 One Additional Feature	Mode with Two Additional Features
B (Basic)	BD (Basic direct)	BDS (BD single-write start)
		BDE (BD external control)
	BC (Basic chaining)	BCE (BC external control)
		BCS (BC single-write start)
Ext (Extended)	ExtD (Extended direct)	ExtDS (ExtD single-write start)
		ExtDE (ExtD external control)
	ExtC (Extended chaining)	ExtCE (ExtC external control)
		ExtCS (ExtC single-write start)

Table 14-2 describes bit settings required for each DMA mode of operation

Table 14-2. DMA Mode Field Descriptions

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
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

Table 14-2. DMA Mode Field Descriptions (continued)

Modes with Features	MR*n*[XFE]	MR*n*[CTM]	MR*n*[SRW]	MR*n*[EM]	MR*n*[CDSM/SWSM]	MR*n*[EMS_EN]
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 shows the general DMA operational flow chart.

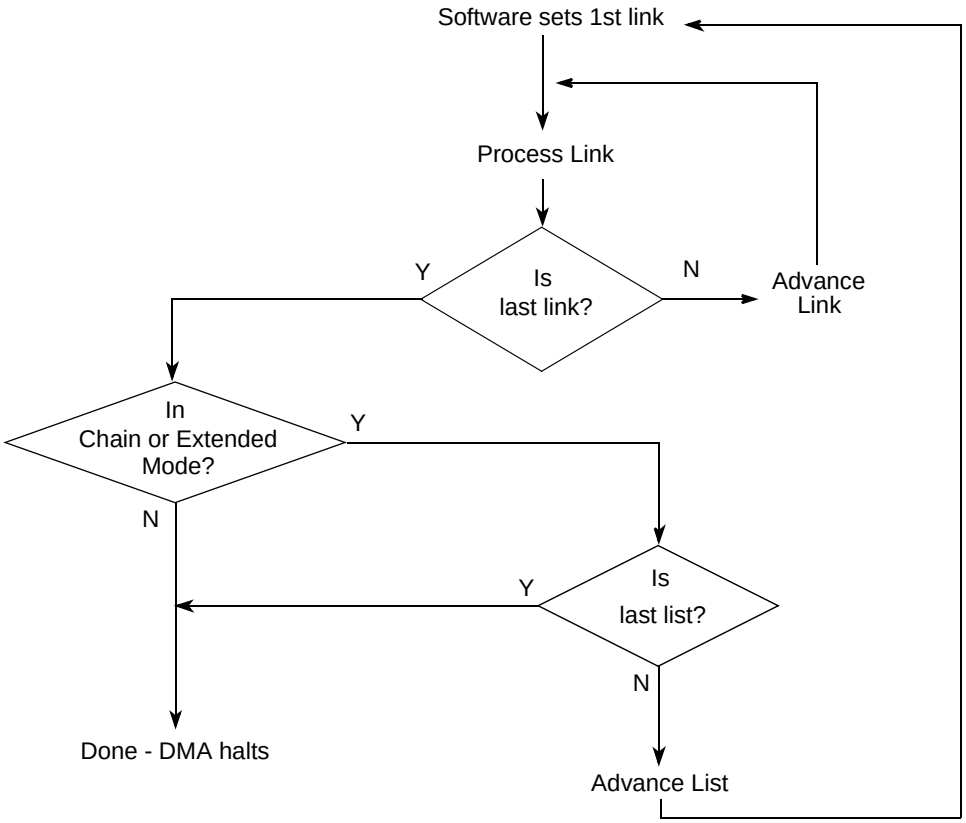


Figure 14-2. DMA Operational Flow Chart

14.2 External Signal Descriptions

This section describes the DMA signals.

14.2.1 Signal Overview

Figure 14-3 summarizes the DMA controller signals.

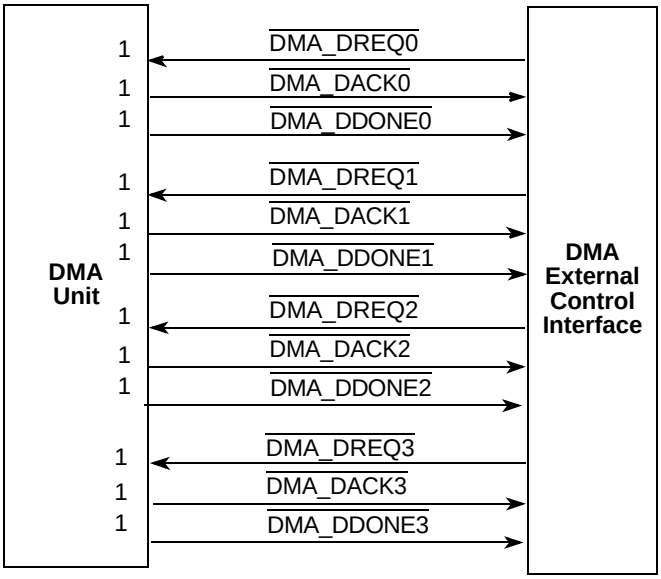


Figure 14-3. DMA Signal Summary

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 17.4.1.10, “Alternate Function Signal Multiplex Control Register (PMUXCR).”

14.2.2 Detailed Signal Descriptions

Table 14-3 describes the DMA signals.

Table 14-3. DMA Signals—Detailed Signal Descriptions

Signal	I/O	Description	
$\overline{\text{DMA\_DREQ}n}$ DMA request	I	DMA request. The DMA request signal indicates the start of a DMA transfer or a restart from a paused request. The falling edge of $\overline{\text{DMA\_DREQ}n}$ causes MR*n*[CS] to be set, thereby activating the corresponding DMA channel.	
		State Meaning	Asserted—Assertion of $\overline{\text{DMA\_DREQ}n}$ while $\overline{\text{DMA\_DACK}n}$ is negated causes a new transfer to start OR resumes a paused transfer if the EMP_EN bit is set. Assertion while $\overline{\text{DMA\_DACK}n}$ is asserted results in an illegal condition Negated—Negation while $\overline{\text{DMA\_DACK}n}$ is asserted has no effect. Negation before the assertion of $\overline{\text{DMA\_DACK}n}$ results in an illegal condition
		Timing	Assertion—Can be asserted asynchronously Negation— Must remain asserted at least until the assertion of the corresponding $\overline{\text{DMA\_DACK}n}$
$\overline{\text{DMA\_DACK}n}$	O	DMA acknowledge. Indicates that a DMA transfer is currently in progress	
		State Meaning	Asserted—Indicates that a DMA transfer is currently in progress. Asserted after the assertion of $\overline{\text{DMA\_DREQ}}$ 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—Asynchronous assertion; asserted for more than three system clocks Negation—Asynchronous negation; negated for more than three system clocks
$\overline{\text{DMA\_DDONE}n}$	O	DMA done. Indicates that a DMA transfer is complete.	
		State Meaning	Asserted—Indicates transfer completion. SRn[CB] is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface. Negated—Indicates that the current transfer is in process
		Timing	Assertion—Always asserts asynchronously after the negation of the final $\overline{\text{DMA\_DACK}n}$ to indicate completion of a transfer. For a paused transfer, $\overline{\text{DMA\_DDONE}n}$ is asserted asynchronously after the negation of the final $\overline{\text{DMA\_DACK}n}$. Negation—Negated asynchronously after the assertion of $\overline{\text{DMA\_DREQ}n}$ 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 shows the address map of DMA register space.

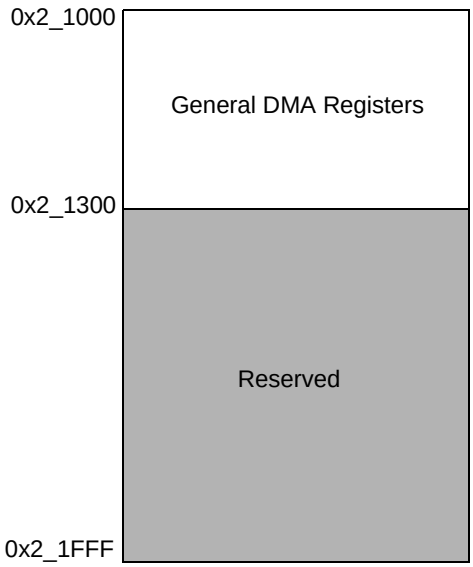


Figure 14-4. DMA Register Space (Memory-Mapped)

Table 14-4 lists the DMA registers.

Table 14-4. DMA Register Summary

Offset	Register	Access	Reset	Section/Page
General Registers				
0x2_1100	MR0—DMA mode register 0	R/W	0x0000_0000	14.3.2.1/14-10
0x2_1104	SR0—DMA status register 0	Special	0x0000_0000	14.3.2.2/14-13
0x2_1108	Reserved	—	—	—
0x2_110C	CLNDAR0—DMA current link descriptor address register 0	R/W	0x0000_0000	14.3.2.3/14-14
0x2_1110	SATR0—DMA source attributes register 0	R/W	0x0000_0000	14.3.2.4/14-16
0x2_1114	SAR0—DMA source address register 0	R/W	0x0000_0000	14.3.2.5/14-18
0x2_1118	DATR0—DMA destination attributes register 0	R/W	0x0000_0000	14.3.2.6/14-19
0x2_111C	DAR0—DMA destination address register 0	R/W	0x0000_0000	14.3.2.7/14-21
0x2_1120	BCR0—DMA byte count register 0	R/W	0x0000_0000	14.3.2.8/14-23
0x2_1124	Reserved	—	—	—
0x2_1128	NLNDAR0—DMA next link descriptor address register 0	R/W	0x0000_0000	14.3.2.9/14-23
0x2_112C–0x2_1130	Reserved	—	—	—

Table 14-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x2_1134	CLSDAR0—DMA current list alternate base descriptor address register 0	R/W	0x0000_0000	14.3.2.10/14-24
0x2_1138	Reserved	—	—	—
0x2_113C	NLSDAR0—DMA next list descriptor address register 0	R/W	0x0000_0000	14.3.2.11/14-25
0x2_1140	SSR0—DMA source stride register 0	R/W	0x0000_0000	14.3.2.12/14-25
0x2_1144	DSR0—DMA destination stride register 0	R/W	0x0000_0000	14.3.2.13/14-26
0x2_1148–0x2_117C	Reserved	—	—	—
0x2_1180	MR1—DMA mode register 1	R/W	0x0000_0000	14.3.2.1/14-10
0x2_1184	SR1—DMA status register 1	Special	0x0000_0000	14.3.2.2/14-13
0x2_1188	Reserved	—	—	—
0x2_118C	CLNDAR1—DMA current link descriptor address register 1	R/W	0x0000_0000	14.3.2.3/14-14
0x2_1190	SATR1—DMA source attributes register 1	R/W	0x0000_0000	14.3.2.4/14-16
0x2_1194	SAR1—DMA source address register 1	R/W	0x0000_0000	14.3.2.5/14-18
0x2_1198	DATR1—DMA destination attributes register 1	R/W	0x0000_0000	14.3.2.6/14-19
0x2_119C	DAR1—DMA destination address register 1	R/W	0x0000_0000	14.3.2.7/14-21
0x2_11A0	BCR1—DMA byte count register 1	R/W	0x0000_0000	14.3.2.8/14-23
0x2_11A4	Reserved	—	—	—
0x2_11A8	NLNDAR1—DMA next link descriptor address register 1	R/W	0x0000_0000	14.3.2.9/14-23
0x2_11AC–0x2_11B0	Reserved	—	—	—
0x2_11B4	CLSDAR1—DMA current list alternate base descriptor address register 1	R/W	0x0000_0000	14.3.2.10/14-24
0x2_11B8	Reserved	—	—	—
0x2_11BC	NLSDAR1—DMA next list descriptor address register 1	R/W	0x0000_0000	14.3.2.11/14-25
0x2_11C0	SSR1—DMA source stride register 1	R/W	0x0000_0000	14.3.2.12/14-25
0x2_11C4	DSR1—DMA destination stride register 1	R/W	0x0000_0000	14.3.2.13/14-26
0x2_11C8–0x2_11FC	Reserved	—	—	—
0x2_1200	MR2—DMA mode register 2	R/W	0x0000_0000	14.3.2.1/14-10
0x2_1204	SR2—DMA status register 2	Special	0x0000_0000	14.3.2.2/14-13
0x2_1208	Reserved	—	—	—
0x2_120C	CLNDAR2—DMA current link descriptor address register 2	R/W	0x0000_0000	14.3.2.3/14-14

Table 14-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x2_1210	SATR2—DMA source attributes register 2	R/W	0x0000_0000	14.3.2.4/14-16
0x2_1214	SAR2—DMA source address register 2	R/W	0x0000_0000	14.3.2.5/14-18
0x2_1218	DATR2—DMA destination attributes register 2	R/W	0x0000_0000	14.3.2.6/14-19
0x2_121C	DAR2—DMA destination address register 2	R/W	0x0000_0000	14.3.2.7/14-21
0x2_1220	BCR2—DMA byte count register 2	R/W	0x0000_0000	14.3.2.8/14-23
0x2_1224	Reserved	—	—	—
0x2_1228	NLNDAR2—DMA next link descriptor address register 2	R/W	0x0000_0000	14.3.2.9/14-23
0x2_122C–0x2_1230	Reserved	—	—	—
0x2_1234	CLSDAR2—DMA current list alternate base descriptor address register 2	R/W	0x0000_0000	14.3.2.10/14-24
0x2_1238	Reserved	—	—	—
0x2_123C	NLSDAR2—DMA next list descriptor address register 2	R/W	0x0000_0000	14.3.2.11/14-25
0x2_1240	SSR2—DMA source stride register 2	R/W	0x0000_0000	14.3.2.12/14-25
0x2_1244	DSR2—DMA destination stride register 2	R/W	0x0000_0000	14.3.2.13/14-26
0x2_1248–0x2_127C	Reserved	—	—	—
0x2_1280	MR3—DMA mode register 3	R/W	0x0000_0000	14.3.2.1/14-10
0x2_1284	SR3—DMA status register 3	Special	0x0000_0000	14.3.2.2/14-13
0x2_1288	Reserved	—	—	—
0x2_128C	CLNDAR3—DMA current link descriptor address register 3	R/W	0x0000_0000	14.3.2.3/14-14
0x2_1290	SATR3—DMA source attributes register 3	R/W	0x0000_0000	14.3.2.4/14-16
0x2_1294	SAR3—DMA source address register 3	R/W	0x0000_0000	14.3.2.5/14-18
0x2_1298	DATR3—DMA destination attributes register 3	R/W	0x0000_0000	14.3.2.6/14-19
0x2_129C	DAR3—DMA destination address register 3	R/W	0x0000_0000	14.3.2.7/14-21
0x2_12A0	BCR3—DMA byte count register 3	R/W	0x0000_0000	14.3.2.8/14-23
0x2_12A4	Reserved	—	—	—
0x2_12A8	NLNDAR3—DMA next link descriptor address register 3	R/W	0x0000_0000	14.3.2.9/14-23
0x2_12AC–0x2_12B0	Reserved	—	—	—
0x2_12B4	CLSDAR3—DMA current list alternate base descriptor address register 3	R/W	0x0000_0000	14.3.2.10/14-24
0x2_12B8	Reserved	—	—	—
0x2_12BC	NLSDAR3—DMA next list descriptor address register 3	R/W	0x0000_0000	14.3.2.11/14-25

Table 14-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x2_12C0	SSR3—DMA source stride register 3	R/W	0x0000_0000	14.3.2.12/14-25
0x2_12C4	DSR3—DMA destination stride register 3	R/W	0x0000_0000	14.3.2.13/14-26
0x2_12C8–0x2_12FC	Reserved	—	—	—
0x2_1300	DGSR—DMA general status register	Read	0x0000_0000	14.3.2.14/14-26

14.3.2 DMA Register Descriptions

The following sections describe the DMA registers. The majority of these registers are channel-specific and can be identified by one of the four offsets that describe the register.

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 describes the MR*n*.

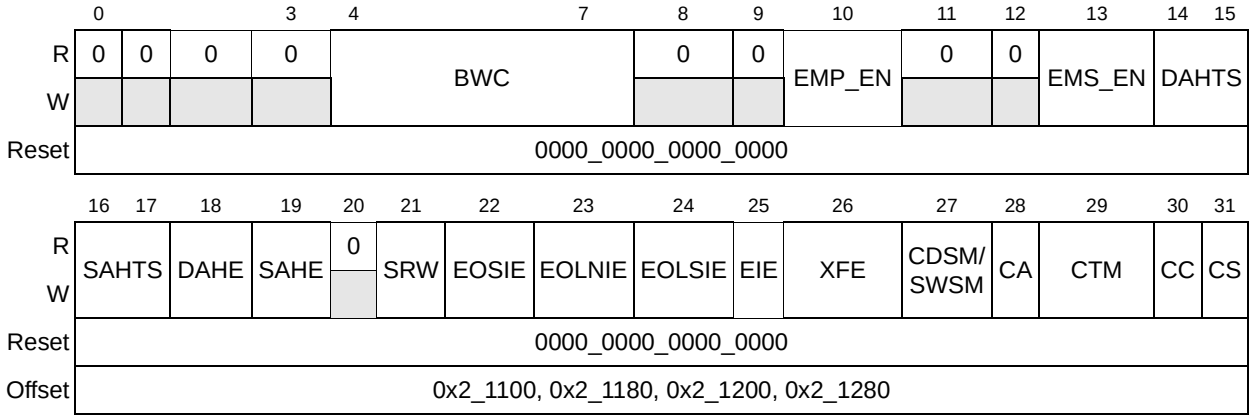


Figure 14-5. DMA Mode Registers (MR*n*)

Table 14-5 describes the MR*n* fields.

Table 14-5. MR*n* Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–7	BWC	Bandwidth/pause control. Defined only if multiple channels are executing transfers concurrently or if MR*n*[EMP_EN] is set. Determines how many bytes a given channel is allowed to transfer before the DMA engine switches to the next channel. The DMA channel pauses during this switch if MR*n*[EMP_EN] is set for this channel. 0000 1 byte 0001 2 bytes 0010 4 bytes 0011 8 bytes 0100 16 bytes 0101 32 bytes 0110 64 bytes 0111 128 bytes 1000 256 bytes 1001 512 bytes 1010 1024 bytes 1011–1110 Reserved 1111 Disable bandwidth sharing to allow uninterrupted transfers from each channel.
8–9	—	Reserved
10	EMP_EN	External master pause enable. Valid only if MR*n*[EMS_EN] is set. 0 Disable the channel from being paused by an external master. 1 Enable the channel to be paused by an external master.
11–12	—	Reserved
13	EMS_EN	External master start enable. This bit does not apply to single-write start modes (direct or chaining). 0 Disable the channel from being started by an external DMA start pin. 1 Enable the channel to be started by an external DMA start pin, which sets MR*n*[CS].
14–15	DAHTS	Destination address hold transfer size. Indicates the transfer size used for each transaction while MR*n*[DAHE] is set. The byte count register must be in multiples of the size and the destination address register must be aligned based on the size. The transfer size assigned to MR*n*[DAHTS] must be equal to or smaller than that assigned to MR*n*[BWC] to avoid undefined behavior. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
16–17	SAHTS	Source address hold transfer size. Indicates the transfer size used for each transaction while MR*n*[SAHE] is set. The byte count register must be in multiples of the size and the source address register must be aligned based on the size. The transfer size assigned to MR*n*[SAHTS] must be equal to or smaller than that assigned to MR*n*[BWC] to avoid undefined behavior. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
18	DAHE	Destination address hold enable 0 Disable destination address hold. 1 Enable the DMA controller to hold the destination address of a transfer to the size specified by MR*n*[DAHTS]. Hardware only supports aligned transfers for this feature.
19	SAHE	Source address hold enable 0 Disable source address hold. 1 Enable the DMA controller to hold the source address of a transfer to the size specified by MR*n*[SAHTS]. Hardware only supports aligned transfers for this feature.

Table 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 CDSM/SWSM causes a write to the destination address register to simultaneously set MR*n*[CS], starting a DMA transfer.
22	EOSIE	End-of-segments interrupt enable 0 Do not generate an interrupt at the completion of a data transfer. CLNDAR*n*[EOSIE] overrides this bit on a link descriptor basis. 1 Generate an interrupt at the completion of a data transfer (That is, SR*n*[EOSI] is set). This bit overrides the CLNDAR*n*[EOSIE].
23	EOLNIE	End-of-links interrupt enable 0 Do not generate an interrupt at the completion of a list of DMA transfers. 1 Generate an interrupt at the completion of a list of DMA transfers (That is, NLNDAR*n*[EOLND] is set).
24	EOLSIE	End-of-lists interrupt enable 0 Do not generate an interrupt at the completion of all DMA transfers. 1 Generate an interrupt at the completion of all DMA transfers (That is, NLNDAR*n*[EOLND] and NLSDAR*n*[EOLSD] are set).
25	EIE	Error interrupt enable 0 Do not generate an interrupt if a programming or transfer error is detected. 1 Generate an interrupt if a programming or transfer error is detected.
26	XFE	Extended features enable 0 Disable the new chaining features. 1 Enable the new chaining features.
27	CDSM/ SWSM	• In chaining mode: Current descriptor start mode/single-write start mode. • In basic mode (MR*n*[XFE] is cleared), setting this bit causes a write to the current link descriptor address register to simultaneously set MR*n*[CS], starting a DMA transfer. • In extended chaining mode (MR*n*[XFE] is set), setting this bit causes a write to the current list descriptor address register to simultaneously set MR*n*[CS], starting a DMA transfer. • In direct mode: Setting this bit and MR*n*[SRW] causes a write to the source address register to simultaneously set MR*n*[CS], starting a DMA transfer. Clearing this bit and setting MR*n*[SRW] causes a write to the destination address register to simultaneously set MR*n*[CS], starting a DMA transfer. This bit must be cleared when MR*n*[SRW] is cleared.
28	CA	Channel abort 0 No effect. 1 Cause the current transfer to be aborted and SR*n*[CB] to be cleared if the channel is busy. The channel remains in the idle state until a new transfer is programmed.
29	CTM	Channel transfer mode 0 Configure the channel in chaining mode. 1 Configure the channel into direct mode. This means that software is responsible for placing all the required parameters into necessary registers to start the DMA process.

Table 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 (CB is cleared). If the channel was halted (CS = 0 and CB = 1), the transfer continues from the point at which it was halted.

14.3.2.2 Status Registers (SR*n*)

The status registers, shown in Figure 14-6, report various DMA conditions during and after a DMA transfer.

	0																					23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TE	0	CH	PE	EOLNI	CB	EOSI	EOLSI
W																														
Reset	0000_0000_0000_0000_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 SR*n*.

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. See Section 14.4.3, “DMA Errors” for additional information.
25	—	Reserved
26	CH	Channel halted. Cleared automatically by hardware if MR*n*[CS] is set again for resuming a halted transfer. 0 Channel is not halted. If software attempts to halt an idle channel (SR*n*[CB] is cleared), this bit will remain 0. 1 DMA transfer was successfully halted by software and it can be resumed.
27	PE	Programming error (Bit reset, write 1 to clear) 0 No programming error detected. 1 A programming error is detected that prevents the DMA transfer from occurring.

Table 14-6. SR*n* Field Descriptions (continued)

Bits	Name	Description
28	EOLNI	End-of-links interrupt. After transferring the last block of data in the last link descriptor, if MR*n*[EOLSIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)
29	CB	Channel busy 0 DMA transfer is finished, an error occurred, or a channel abort occurred. 1 A DMA transfer is currently in progress.
30	EOSI	End-of-segment interrupt. In chaining mode, after finishing a data transfer, if MR*n*[EOSIE] is set or if CLNDAR*n*[EOSIE] is set, this bit gets set and an interrupt is generated. In direct mode, if 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*)

Current link descriptor address registers contain the address of the current link descriptor. In basic chaining mode, shown in Figure 14-7, software must initialize this register to point to the first link descriptor in memory.

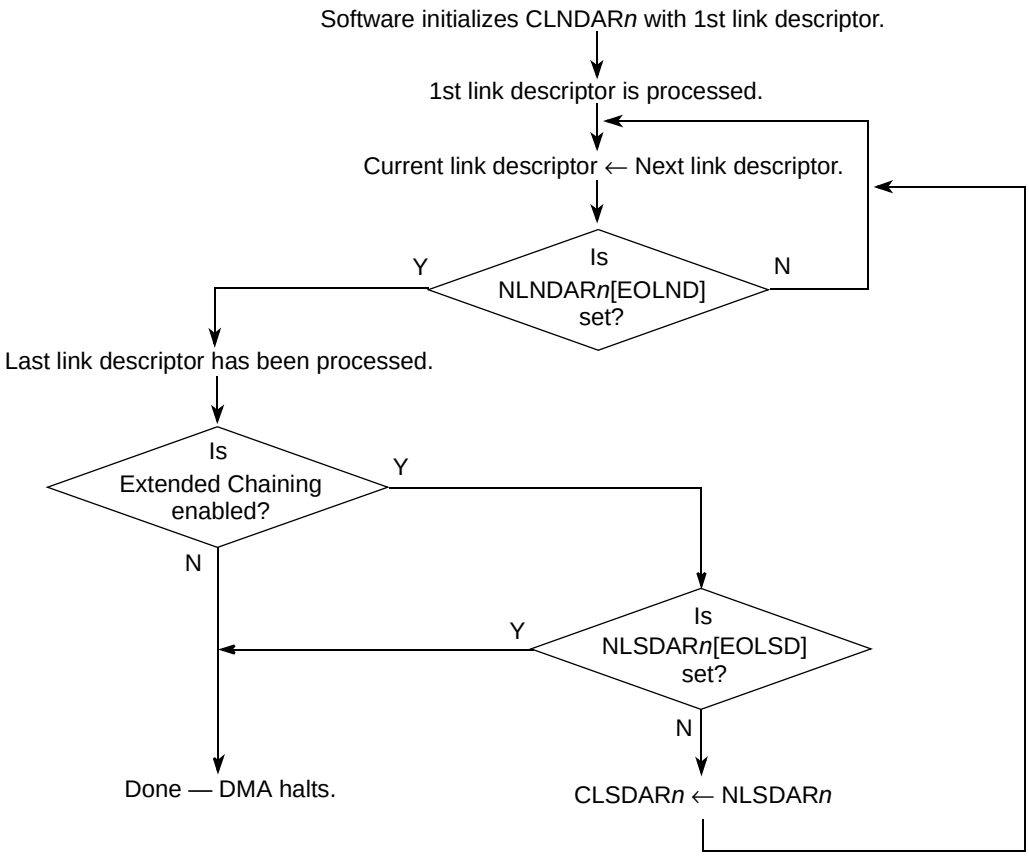


Figure 14-7. Basic Chaining Mode Flow Chart

After the current descriptor is processed, the current link descriptor address register is loaded from the next link descriptor address register and NLNDAR*n*[EOLND] in the next link descriptor address register is examined. If EOLND is zero, the DMA controller reads in the new current link descriptor for processing. If EOLND is set, the last descriptor of the list was just completed. If extended chaining mode is not enabled, all DMA transfers are complete and the DMA controller halts.

If extended chaining mode is enabled, the DMA controller examines the state of NLSDAR*n*[EOLSD] in the next list descriptor address register. If EOLSD is clear, the controller loads the contents of the next list descriptor address register into the current list descriptor address register and reads the new list descriptor from memory. If EOLSD is set, all DMA transfers are complete and the DMA controller halts.

Figure 14-8 shows CLNDAR*n*.

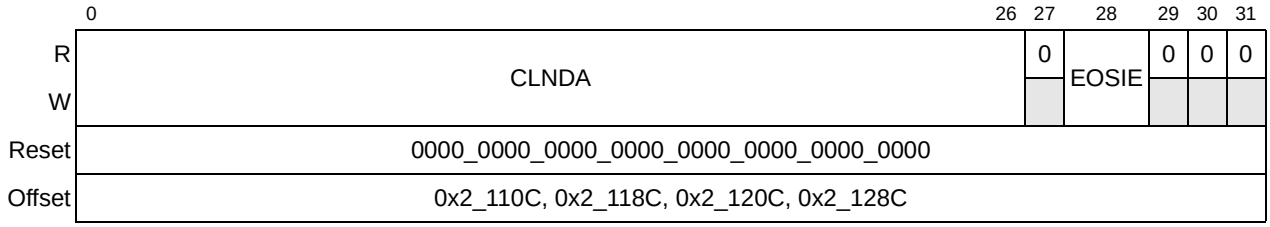


Figure 14-8. Current Link Descriptor Address Registers (CLNDAR*n*)

Table 14-7 describes the fields of the CLNDAR*n*.

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, shown in Figure 14-9, contain the transaction attributes to be used for the DMA operation. Stride mode is enabled by setting SATR*n*[SSME]. Source read transaction type is specified using the SREADTTYPE field.

ATMU bypass mode is enabled by setting SATR*n*[SBPATMU]. ATMU bypass mode is only applicable for accesses to RapidIO. If SBPATMU is set, STRANSINT must be set to RapidIO and attributes that would otherwise come from the ATMU must be specified in this register. If SBPATMU is not set, attributes are derived from local access windows and outbound ATMU registers according to the transaction address.

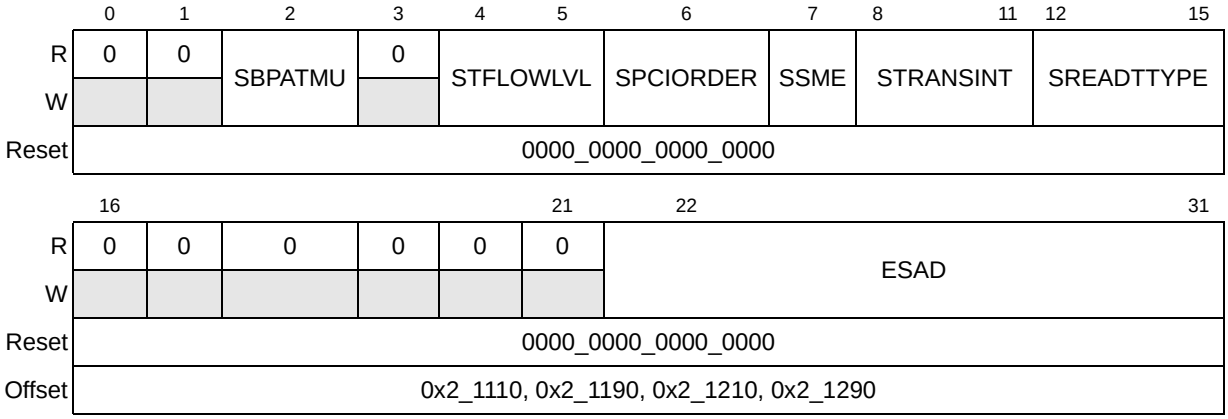


Figure 14-9. Source Attributes Registers (SATR*n*)

Table 14-8 describes the fields of the SATR*n*.

Table 14-8. SATR*n* Field Descriptions

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.
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 SSR*n*.
8–11	STRANSINT	DMA source transaction interface 0000–1011 Reserved 1100 RapidIO interface 1101–1111 Reserved Applicable only to RapidIO interface, while SATR*n*[SBPATMU] is set.

Table 14-8. SATR*n* Field Descriptions (continued)

Bits	Name	Description
12–15	SREADTTYPE	DMA source transaction type. Reserved values will result in a programming error being detected and logged in SR[PE]. • Transaction type to run on RapidIO interface in ATMU bypass mode 0000–0001 Reserved 0010 I/O read home 0011 Reserved 0100 nread 0101–0110 Reserved 0111 maintenance read 1000–1111 Reserved • Transaction type to run on local address space - used even in non-ATMU bypass mode 0000–0001 Reserved 0011 Reserved 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0111 Read, unlock L2 cache line 1000–1111 Reserved
16–21	—	Reserved
22–31	ESAD	Extended source address. ESAD[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, shown in Figure 14-10, contain the address from which the DMA controller reads data. In direct mode, if MR*n*[CDSM/SWSM] and MR*n*[SRW] are set, a write to this register simultaneously sets MR*n*[CS], starting a DMA transfer. Software must ensure that this is a valid address.

	0	31
R W	SAD	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
Offset	0x2_1114, 0x2_1194, 0x2_1214, 0x2_1294	

Figure 14-10. Source Address Registers (SAR*n*)

Table 14-9 describes the field of the SAR*n*.

Table 14-9. SAR*n* Field Descriptions

Bits	Name	Description
0–31	SAD	Source address. This register contains the source address of the DMA transfer. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

14.3.2.5.1 Source Address Registers for RapidIO Maintenance Reads (SAR*n*)

If RapidIO is the source of a transaction, the SAR*n* registers are redefined as shown below. Several options exist for the packet type that can be specified. A maintenance read is one of many possible reads. Maintenance packets have an offset instead of an address. Figure 14-11 describes the SAR*n*.

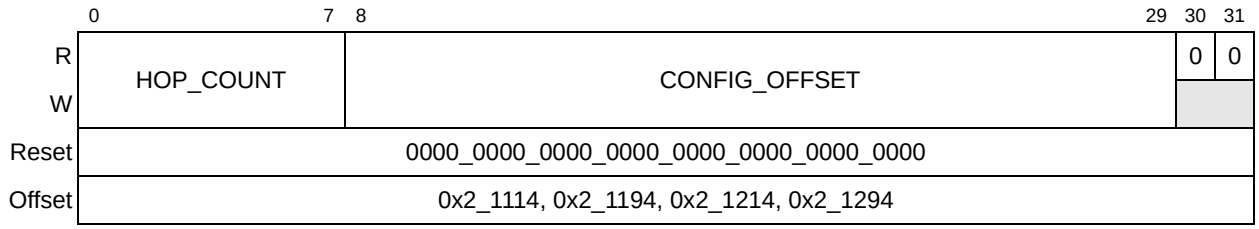


Figure 14-11. Source Address Registers for RapidIO Maintenance Reads (SAR*n*)

Table 14-10 describes the fields of the SAR*n*.

Table 14-10. SAR*n* Field Descriptions

Bits	Name	Description
0–7	HOP_COUNT	Maintenance packet hop_count as defined by the *RapidIO Interconnect Specification 1.2*
8–29	CONFIG_OFFSET	Maintenance packet word offset as defined by the *RapidIO Interconnect Specification 1.2*
30–31	—	Reserved

14.3.2.6 Destination Attributes Registers (DATR*n*)

The destination attributes registers, shown in Figure 14-12, contain the transaction attributes for the DMA operation. Stride mode is enabled by setting DATR*n*[DSME]. Destination write transaction type is specified using the DWRITETTYPE field.

ATMU bypass mode, which is applicable only for accesses to RapidIO, is enabled by setting DATR*n*[DBPATMU]. If DBPATMU is set, DTRANSINT must be set to RapidIO and attributes that would otherwise come from the ATMU must be specified in this register.

If DBPATMU is zero, the target interface is derived from the local access ATMU mapping and the transaction is obtained from the value specified in DWRITETTYPE using the local address space category.

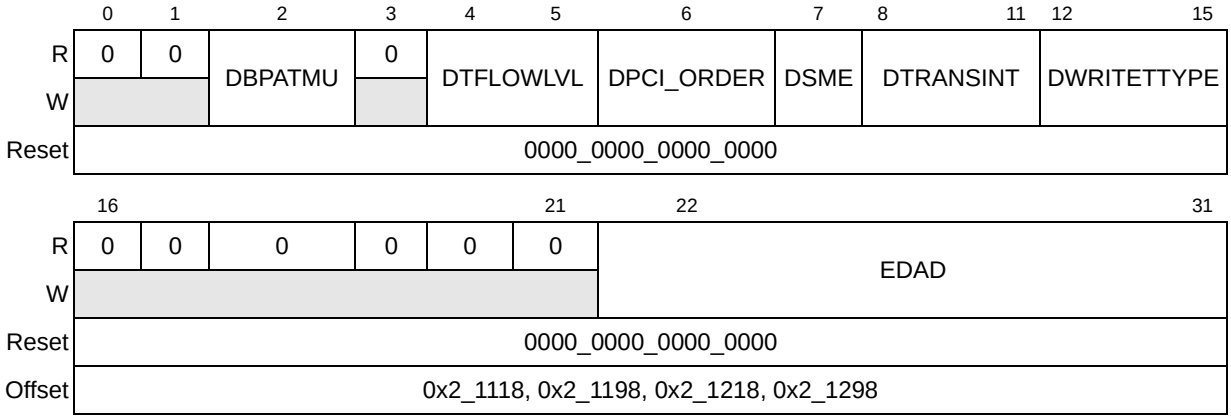


Figure 14-12. Destination Attributes Registers (DATR*n*)

Table 14-11 describes the fields of the 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 setting DSME and setting the desired stride size and distance in DSR*n*.
8–11	DTRANSINT	DMA destination transaction interface 0000–1011 Reserved 1100 RapidIO interface 1101–1111 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. Reserved values will result in a programming error being detected and logged in SR[PE]. • Transaction type to run on RapidIO interface in ATMU bypass mode 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 of size 8, 16, 32, 64, 128 or 256 bytes. (Other message sizes produce boundedly undefined behavior.) 0111 MAINTENANCE write 1000–1111 Reserved • Transaction type to run on local address space—Used even in non-ATMU bypass mode 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
16–21	—	Reserved
22–31	EDAD	Extended destination address. Applicable only for RapidIO interface with DATR*n*[DBPATMU] set. EDAD[0–7] represents the RapidIO target ID. EDAD[8–9] is defined as follows, subject to the transaction type: Message: EDAD[8–9] represents the value for the mbox field in the message packet. Other: EDAD[8–9] represents the two high-order bits of the local device offset.

14.3.2.7 Destination Address Registers (DAR*n*)

The destination address registers, shown in Figure 14-13, contain the addresses to which the DMA controller writes data.

In direct mode, if MR*n*[SRW] is set and MR*n*[CDSM/SWSM] is cleared, a write to this register simultaneously sets MR*n*[CS], starting a DMA transfer. Software must ensure that this is a valid address.

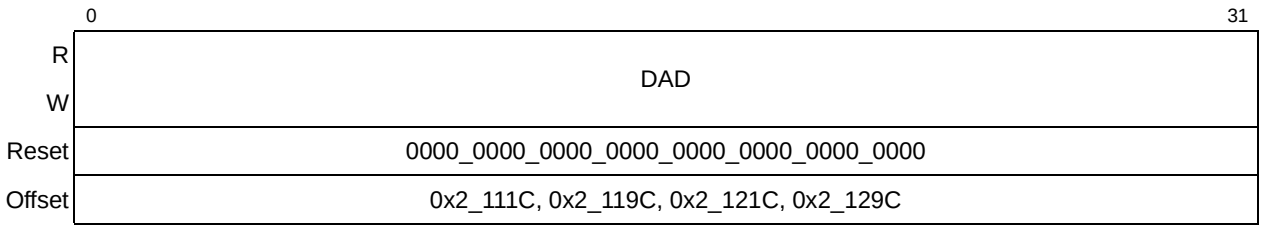


Figure 14-13. Destination Address Registers (DAR*n*)

Table 14-12 describes the field of the DAR*n*.

Table 14-12. DAR*n* Field Descriptions

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 unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

14.3.2.7.1 Destination Address Registers for RapidIO Maintenance Writes (DAR*n*)

If RapidIO is the destination of a transaction, the DAR*n* registers are redefined as shown below. Several options exist for the transaction type that can be specified. There are a number of noncoherent write and flush types for address-based write transactions, and message types for port-based write transactions.

Maintenance packets have an offset instead of an address. Figure 14-14 shows the DAR*n*.

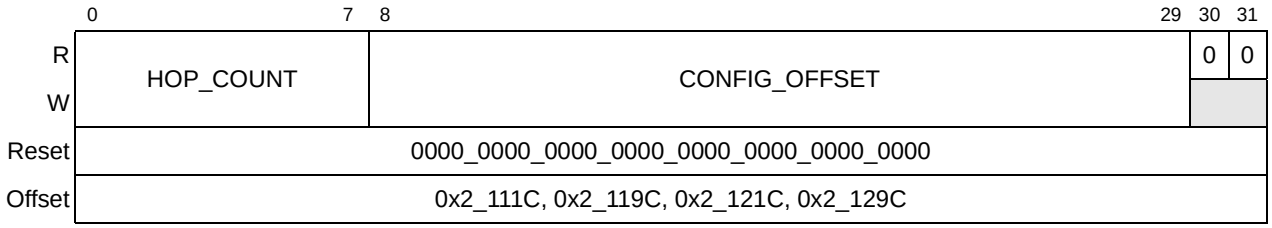


Figure 14-14. Destination Address Registers for RapidIO Maintenance Writes (DAR*n*)

Table 14-13 describes the fields of the DAR*n*.

Table 14-13. DAR*n* Field Descriptions

Bits	Name	Description
0–7	HOP_COUNT	Maintenance packet hop count as defined by the *RapidIO Interconnect Specification 1.2*
8–29	CONFIG_OFFSET	Maintenance packet word offset as defined by the *RapidIO Interconnect Specification 1.2*
30–31	—	Reserved

14.3.2.8 Byte Count Registers (BCR*n*)

The byte count register, shown in Figure 14-15, contains the number of bytes to transfer.

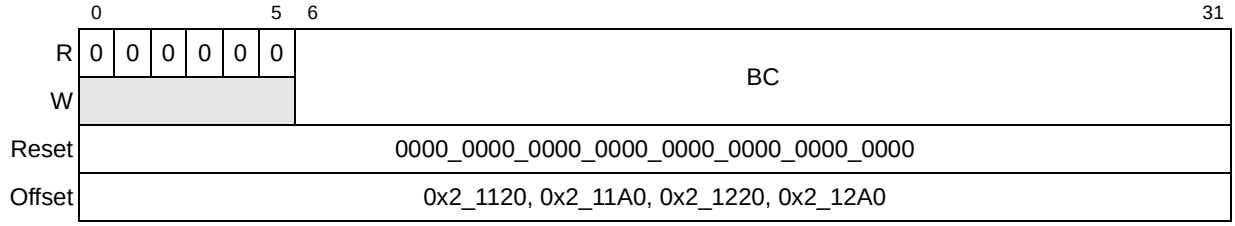


Figure 14-15. Byte Count Registers (BCR*n*)

Table 14-14 describes the fields of the 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 2^{26} – 1 Bytes.

14.3.2.9 Next Link Descriptor Address Registers (NLNDAR*n*)

The next link descriptor address registers, shown in Figure 14-16, contain the address for the next link descriptor in memory. Contents transferred to the current descriptor address registers become effective for the current transfer in basic and extended chaining modes.

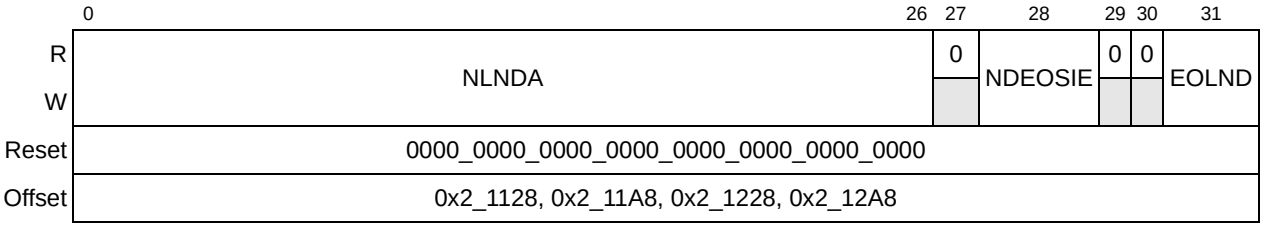


Figure 14-16. Next Link Descriptor Address Registers (NLNDAR*n*)

Table 14-15 describes the fields of the NLNDAR*n* registers.

Table 14-15. 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

Table 14-15. NLNDAR*n* Field Descriptions (continued)

Bits	Name	Description
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, shown in Figure 14-17, contain the current address of the list descriptor in memory in extended chaining mode.

In extended chaining mode, software must initialize CLSDAR*n* to point to the first list descriptor in memory. After finishing the last link descriptor in the current list, the DMA controller loads the contents of the next list descriptor address register into the current list descriptor address register. If NLSDAR*n*[EOLSD] in the next list descriptor address register is clear, the DMA controller reads the new current list descriptor from memory to process that list. If EOLSD in the next list descriptor address register is set and the last link in the current list is finished all DMA transfers are complete.

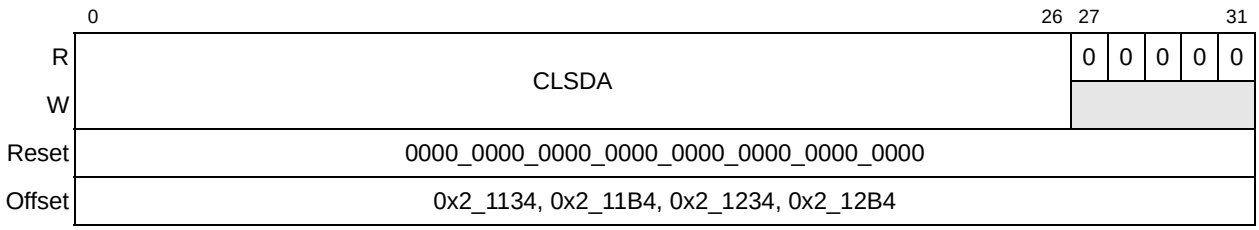


Figure 14-17. Current List Descriptor Address Registers (CLSDAR*n*)

Table 14-16 describes the fields of the CLSDAR*n*.

Table 14-16. CLSDAR*n* Field Descriptions

Bits	Name	Description
0–26	CLSDA	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.
27–31	—	Reserved

14.3.2.11 Next List Descriptor Address Registers (NLSDAR*n*)

The next list descriptor address register, shown in Figure 14-18, 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.

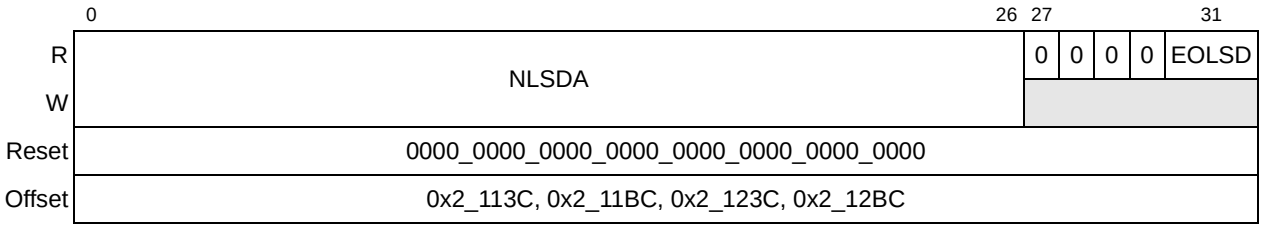


Figure 14-18. Next List Descriptor Address Registers (NLSDAR*n*)

Table 14-17 describes the fields of the NLSDAR*n*.

Table 14-17. 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.
27–30	—	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, shown in Figure 14-19, 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.

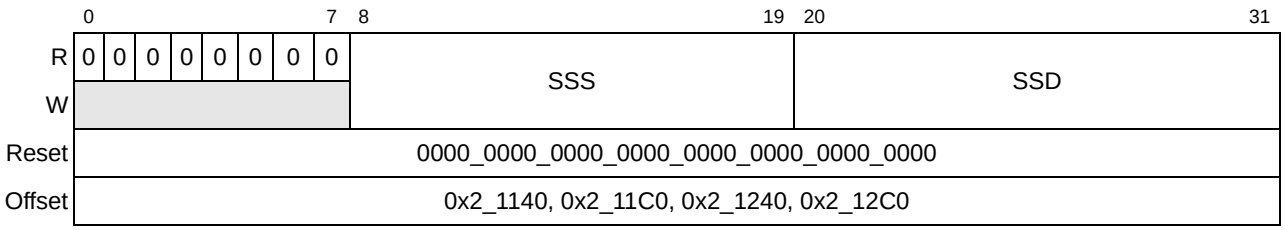


Figure 14-19. Source Stride Registers (SSR*n*)

Table 14-18 describes the fields of the SSR*n*.

Table 14-18. 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 describes the DSR*n*.

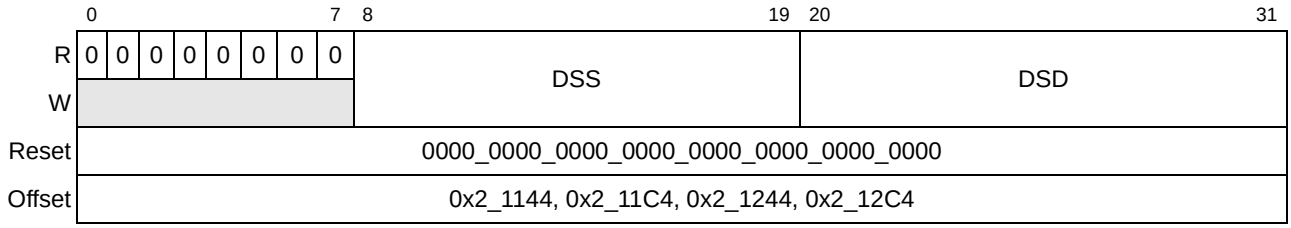


Figure 14-20. Destination Stride Registers (DSR*n*)

Table 14-19 describes the fields of the DSR*n*.

Table 14-19. 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 describes the DGSR.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TE0	0	CH0	PE0	EOLNI0	CB0	EOSI0	EOLSI0	TE1	0	CH1	PE1	EOLNI1	CB1	EOSI1	EOLSI1
W																
Reset	0000_0000_0000_0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TE2	0	CH2	PE2	EOLNI2	CB2	EOSI2	EOLSI2	TE3	0	CH3	PE3	EOLNI3	CB3	EOSI3	EOLSI3
W																
Reset	0000_0000_0000_0000															
Offset	0x2_1300															

Figure 14-21. DMA General Status Register (DGSR)

Table 14-20 describes the fields of the DGSR.

Table 14-20. 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	PE0	Programming error, Channel 0
4	EOLNI0	End-of-links interrupt, Channel 0
5	CB0	Channel busy, Channel 0
6	EOSI0	End-of-segment interrupt, Channel 0
7	EOLSI0	End-of-lists/direct interrupt, Channel 0
8	TE1	Transfer error, Channel 1 0 Normal operation 1 An error condition occurred during the DMA transfer.
9	—	Reserved
10	CH1	Channel halted, Channel 1
11	PE1	Programming error, Channel 1
12	EOLNI1	End-of-links interrupt, Channel 1
13	CB1	Channel busy, Channel 1
14	EOSI1	End-of-segment interrupt, Channel 1
15	EOLSI1	End-of-lists/direct interrupt, Channel 1

Table 14-20. DGSR Field Descriptions (continued)

Bits	Name	Description
16	TE2	Transfer error, Channel 2 0 Normal operation 1 An error condition occurred during the DMA transfer.
17	—	Reserved
18	CH2	Channel halted, Channel 2
19	PE2	Programming error, Channel 2
20	EOLNI2	End-of-links interrupt, Channel 2
21	CB2	Channel busy, Channel 2
22	EOSI2	End-of-segment interrupt, Channel 2
23	EOLSI2	End-of-lists/direct interrupt, Channel 2
24	TE3	Transfer error, Channel 3 0 Normal operation 1 An error condition occurred during the DMA transfer.
25	—	Reserved
26	CH3	Channel halted, Channel 3
27	PE3	Programming error, Channel 3
28	EOLNI3	End-of-links interrupt, Channel 3
29	CB3	Channel busy, Channel 3
30	EOSI3	End-of-segment interrupt, Channel 3
31	EOLSI3	End-of-lists/direct interrupt, Channel 3

14.4 Functional Description

This section describes the function of the the DMA controller.

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, 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 MR*n*, the channel enters a paused state after transferring the data described in the bandwidth control. External control can restart the channel from a paused state.

The DMA controller is designed to support RapidIO transaction types, including various priority level support. The DMA controller offers additional features from previous generations in which the reads can be mapped to globally coherent (IO_READ), non-coherent (NREAD), or maintenance reads. In addition, the writes can be mapped to coherent (flush with data) and non-coherent (NWRITE, NWRITE_R) writes, messages, and maintenance writes.

The DMA programming model permits software to program each DMA engine independently to interrupt on completed segment, chain, or error. It also provides the capability for software to resume the DMA engine from a hardware halted condition by setting the channel continue bit, MR*n*[CC]. See Table 14-21 on page 14-36 for more complete descriptions of the channel states and state transitions.

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 SAR*n*, SATR*n*, DAR*n*, DATR*n*, and BCR*n* registers. The DMA transfer is started when MR*n*[CS] is set. Software is expected to program all the appropriate registers before setting MR*n*[CS] to a 1. The transfer is finished after all the bytes specified in the

byte count register have been transferred or if an error condition occurs. The sequence of events to start and complete a transfer in basic direct mode is as follows:

1. Poll the channel state (see Table 14-21), to confirm that the specific DMA channel is idle.
2. Initialize SAR*n*, SATR*n*, DAR*n*, DATR*n* and BCR*n*.
3. Set the mode register channel transfer mode bit, MR*n*[CTM], to indicate direct mode. Other control parameters may also be initialized in the mode register.
4. Clear then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after the transfer is finished, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if MR*n*[EOSIE] is set.

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 SATR*n*, DATR*n*, and BCR*n* registers. Setting MR*n*[SRW] configures the DMA controller to begin the DMA transfer either when SAR*n* is written or when DAR*n* is written, determined by the state of MR*n*[CDSM/SWSM]. Writing to SAR*n* initiates the DMA transfer if MR*n*[CDSM/SWSM] is set. Writing to DAR*n* initiates the DMA transfer if MR*n*[CDSM/SWSM] is cleared. The DMA controller automatically sets the channel start bit, MR*n*[CS]. Software is expected to program all the appropriate registers before writing the source or destination address registers. The transfer is finished after all the bytes specified in the byte count register have been transferred or if an error condition occurs. The sequence of events to start and complete a transfer in single-write start basic direct mode is as follows:

1. Poll the channel state (see Table 14-21), to confirm that the specific DMA channel is idle.
2. Initialize the source attributes (SATR*n*), DATR*n*, and BCR*n* registers.
3. Set the mode register channel transfer mode bit, MR*n*[CTM], and the single-write start direct mode bit, MR*n*[SRW]. Other control parameters may also be initialized in the mode register. Set MR*n*[CDSM/SWSM] for transfers started using SAR*n*. Clear MR*n*[CDSM/SWSM] for transfers started using the DAR*n*.
4. A write to the source or destination address register starts the DMA transfer and automatically sets MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after the transfer is finished, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if MR*n*[EOSIE] is set.

14.4.1.1.3 Basic Chaining Mode

In basic chaining mode, software must first build link descriptor segments in memory. Then the current link descriptor address register must be initialized to point to the first descriptor in memory. The DMA controller loads descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded for the segment. After the current segment is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer. The transfer is finished if the current link descriptor is the last one in memory or if an error condition occurs. The sequence of events to start and complete a transfer in chaining mode is as follows:

1. Build link descriptor segments in memory. See Section 14.4.4, “DMA Descriptors,” for more information on descriptors.
2. Poll the channel state (see Table 14-21), to confirm that the specific DMA channel is idle.
3. Initialize 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. See Section 14.4.4, “DMA Descriptors,” for more information on descriptors.
3. Poll the channel state (see Table 14-21), to confirm that the specific DMA channel is idle.
4. Initialize 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 SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

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 SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

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 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. See Section 14.4.4, “DMA Descriptors,” for more information on descriptors.

2. Poll the channel state (see Table 14-21), to confirm that the specific DMA channel is idle.
3. Initialize 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 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 channel state (see Table 14-21), to confirm that the specific DMA channel is idle.
4. Initialize the current list descriptor address register to point to the first list descriptor segment in memory. This write automatically causes the DMA controller to begin the list descriptor fetch and set MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

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. The falling edge of $\overline{\text{DMA\_DREQ}}$ 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 that the DMA engine has completed the transfer. SRn[CB] is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface.

Detailed descriptions of the external control interface are in Table 14-4. The timing diagram of the external control interface is shown in Figure 14-22.

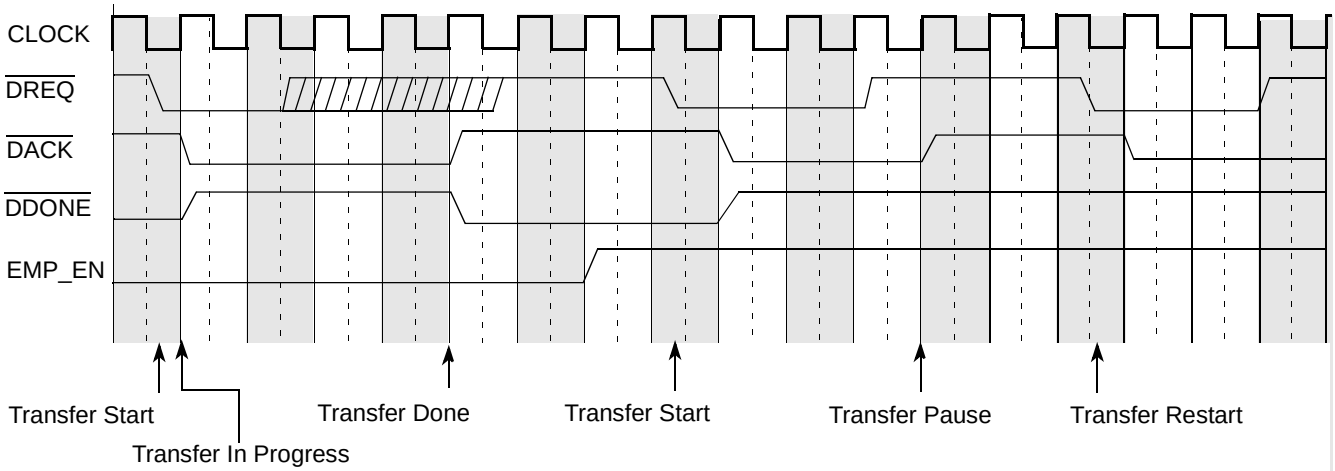


Figure 14-22. External Control Interface Timing

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 halted state while software continues to build other descriptors in memory. Software can then set CC to force hardware to continue where it left off. Channel continue is only meaningful for chaining modes, not direct mode.

If CC is set by software while the channel is busy with a transfer, the DMA controller finishes all transfers until it reaches the EOLND in basic mode or EOLSD in extended mode. The DMA controller then refetches the last link descriptor in basic mode, or the last list descriptor in extended mode and clears the channel continue bit. If EOLND or EOLSD is still set for their respective modes, the DMA controller remains in the idle state. If EOLND or EOLSD is not set, the DMA controller continues the transfer by refetching the new descriptor.

If CC is set by software while the channel is not busy with a transfer, the DMA controller refetches the last link descriptor in basic mode, or the last list descriptor in extended mode and clears the channel continue bit. If EOLND or EOLSD is still set for their respective modes, the DMA controller remains in the idle state. If the EOLND or EOLSD bits are not set, the DMA controller continues the transfer by refetching the new descriptor.

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 EOLND is zero, the next link descriptor address is copied into CLNDAR*n* and the channel continues with another descriptor fetch of the current link descriptor address. As a result, two link descriptor fetches always exist after channel continue before starting the first transfer.

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 zero-to-one transition of MR*n*[CA], it finishes the current sub-block transfer and halts all further activity. The controller then waits for all previously initiated transfers from the specified channel to drain and clears SR*n*[CB]. Successful completion of a software initiated abort request can be recognized by MR*n*[CA] being set and SR*n*[CB] being cleared. Obviously, if the controller was already halted because of an error condition (SR*n*[TE] is set), or the channel has completed all transfers, then SR*n*[CB] being cleared may not signify that the controller entered a halt state due to the abort request.

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 Kbyte at a time when no other channel is active.

14.4.1.7 Channel State

Table 14-21 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 14-21. 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 transfer (byte count > 0), or transfer complete (byte count = 0)
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 repeats until the amount of data transferred equals the transfer size.

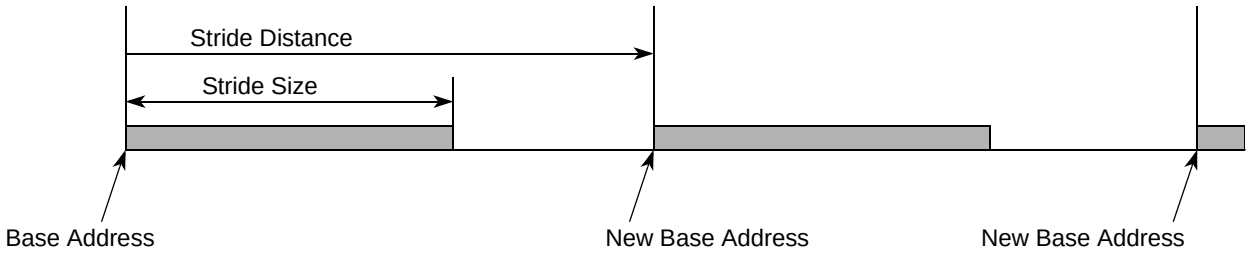


Figure 14-23. Stride Size and Stride Distance

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 (uncorrectable ECC errors on memory accesses, parity errors on local bus or PCI, address mapping errors, for example), the DMA halts by setting SR*n*[TE] and generates an interrupt if MR*n*[EIE] is set. On a programming error, the DMA sets SR*n*[PE] and generates an interrupt if MR*n*[EIE] is set. The DMA controller detects the following programming errors:

- Transfer started with a byte count of zero.
- Stride transfer started with a stride size of zero.
- Transfer started with a priority of three.
- Illegal type—Defined by SATR*n*[SREADTTYPE] and DATR*n*[DWRITETTYPE]. Used for the transfer.
- Invalid interface—Defined by SATR*n*[STRANSINT] and DATR*n*[DTRANSINT]. Used for the transfer when in ATMU bypass mode.

14.4.4 DMA Descriptors

The DMA engine recognizes list descriptors and link descriptors. List descriptors connect lists of link descriptors. Link descriptors describe the DMA activity that is to take place. DMA descriptors are built in either local or remote memory and are connected by the next descriptor fields. Only link descriptors contain information for the DMA controller to transfer data. Software must ensure that each descriptor is 32-byte aligned. The last link descriptor in the last list in memory sets the EOLND bit in the next link descriptor; the next list descriptor fields indicating that these are the last descriptors in memory. Software initializes the current list descriptor address register to point to the first list descriptor in memory. The DMA controller traverses through the descriptor lists until the last link descriptor is met. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor. Link and list descriptor fetches always snoop the local memory space.

NOTE

Software must ensure that each descriptor is aligned on a 32-byte boundary.

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 14-24. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor. Table 14-22 summarizes the DMA list descriptors.

Table 14-22. DMA List 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 14-23 summarizes the DMA link descriptors.

Table 14-23. DMA List 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.

Figure 14-24 describes the DMA transaction flow.

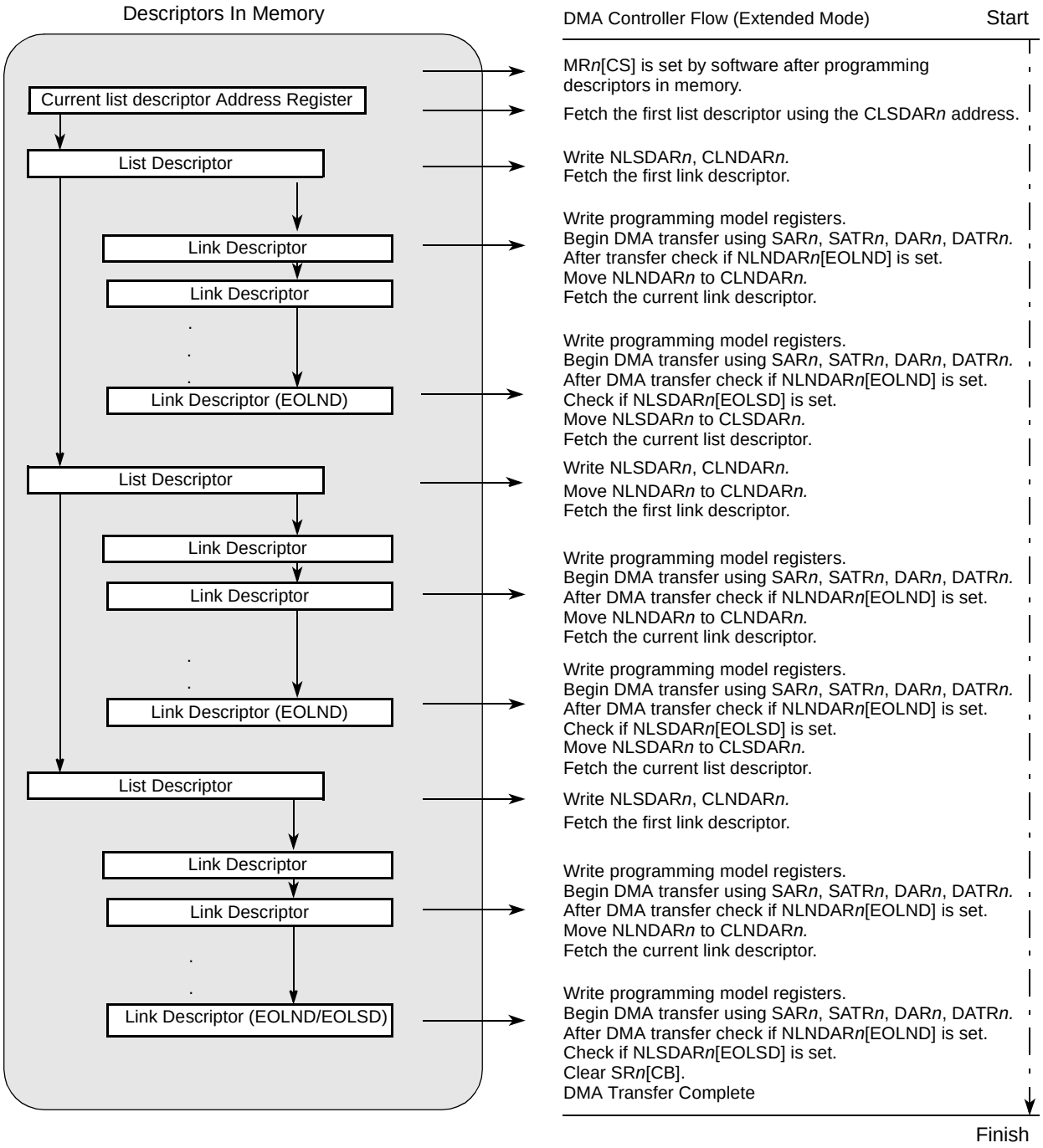


Figure 14-24. DMA Transaction Flow with DMA Descriptors

Figure 14-25 describes the format of the list descriptors.

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 14-25. List Descriptor Format

Figure 14-26 describes the format of the link descriptors.

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 14-26. Link Descriptor Format

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 as follows:

- Setting the source or destination priority level (STFLOWLVL or DTFLOWLVL) to a value of three (0b11) is considered a programming error.
- All interface capabilities from where descriptors are being fetched must support read sizes 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 SAHTS.
- If MR*n*[DAHE] is set, the destination interface transfer size capability must be greater than or equal to MR*n*[DAHTS]. The destination address must be aligned to the size specified by DAHTS.
- Destination striding is not supported if MR*n*[DAHE] is set and source striding is not supported if MR*n*[SAHE] is set.
- If the DMA is programmed to send SWRITEs over RapidIO, the programmer must ensure that the destination address is double-word aligned and that the byte count is a double-word multiple.
- If the DMA is programmed to send messages over RapidIO, the programmer must ensure that the message length (BCR*n*[BC]) is 8, 16, 32, 64, 128 or 256 bytes. This can be achieved by setting the byte count register (BCR) to a power of 2 value equal to 8 or greater.
- Striding does not work if the destination transaction type is MESSAGE (DATR[DWRITETTYPE] = 0x0110) because messages have no memory addresses. As well, destination address hold should be disabled (MR*n*[DAHE] is cleared) unless the destination address hold transfer size indicates an 8-byte message (MR*n*[DAHTS] = 0x11). Software is responsible for disabling striding and DAHE, in this case, and for ensuring that the bandwidth control is large enough to support the desired message size. Failure to adhere to these restrictions results in boundedly undefined behavior.
- When DMA is used to issue maintenance reads and writes in bypass mode, the sum of the starting offset field in DAR and the byte count must not cause the offset to rollover. Failure to adhere to this restriction results in boundedly undefined behavior.

14.5 DMA System Considerations

Figure 14-27 shows 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 (perhaps during system boot, for example).

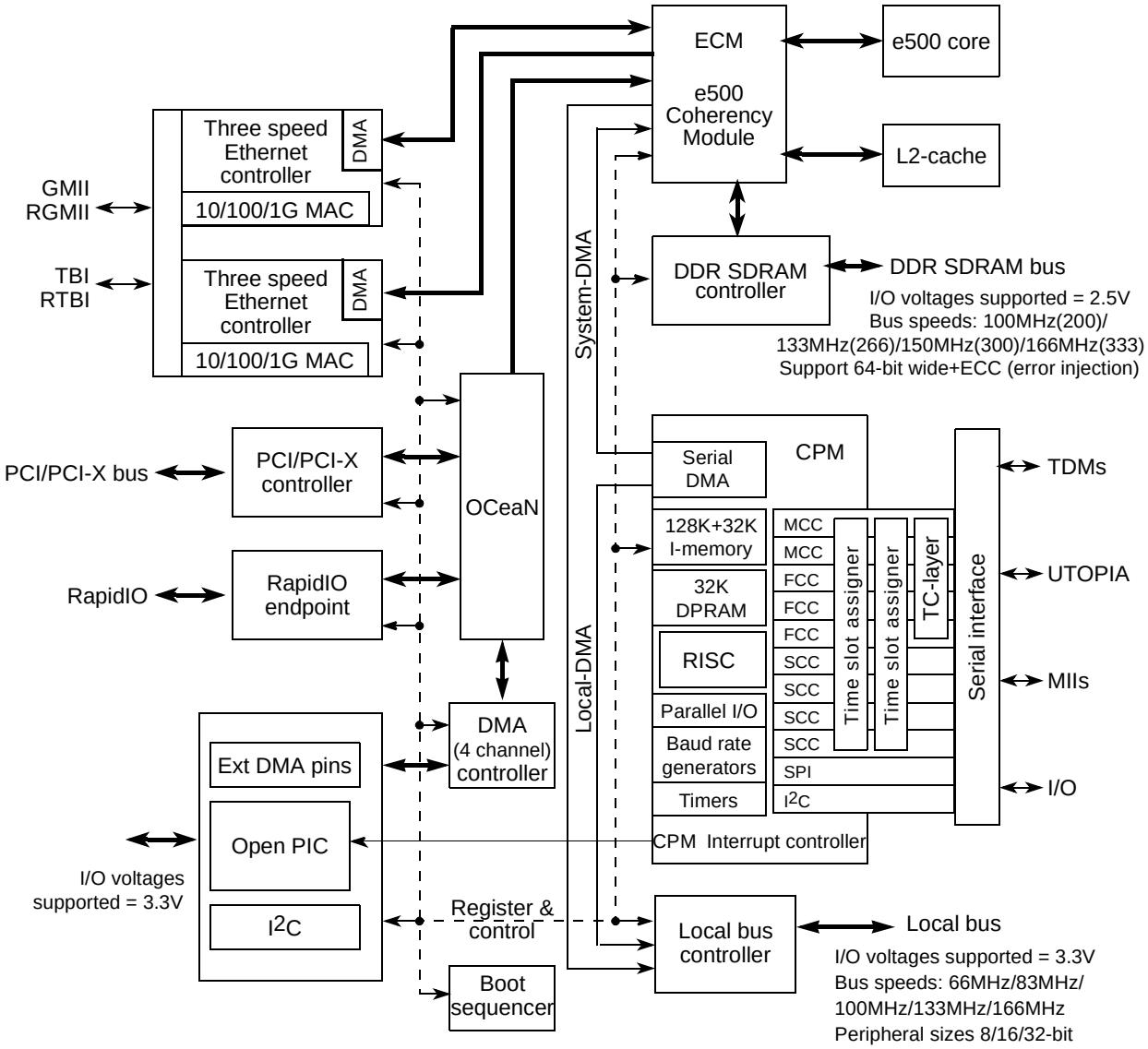


Figure 14-27. DMA Data Paths

Table 14-24 lists all of 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.

Table 14-24. DMA Paths

DMA Controllers		On-Chip Targets			Off-Chip Targets				
		L2	CPM DPRAM	Config[1] Registers	DDR	Local Bus	PCI/ PCI-X	RapidIO	Ethernet Buffers
On-Chip	CPM[2]	Y	NR	NS	Y	Y	Y	Y	Y
	Ethernet[2]	Y	NS	NS	Y	Y	Y	Y	Y
	4 Channel[3]	Y	Y	NR	Y	Y	Y	Y	Y
Off-Chip	PCI/PCI-X Controller	Y	Y	NS	Y	Y	–	Y	Y
	RapidIO Controller	Y	Y	NS	Y	Y	Y	–	Y

1 Includes I^2C data register.
2 Captive resource. Not available to external masters.
3 Can serve external masters.

14.5.1 Unusual DMA Scenarios

The following is a description of unusual DMA paths including explanations of why some functional blocks cannot serve as DMA targets. The following topics are addressed:

- Transaction initiators (masters)
- DMA targets, that is, data sources or destinations
- Transparency of the bus controllers to DMA transactions
- What is useful as opposed to what is possible. For example, any register can be addressed through 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 cannot be a direct DMA target because it cannot be directly addressed by software. However, DMA access into the L1 cache occurs indirectly if a block of memory that is cached in the L1 is specified as the DMA target. This effect is deterministic if the target memory block was locked into the L1 with cache locking instructions.

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 cannot serve as DMA targets because they have no suitable internal memory for this purpose. The Ethernet controllers have dedicated DMA channels to move data between the external transmit and receive buffers and the internal packet buffer. This dedicated channel is the only DMA service to the internal packet buffers.

However, Ethernet ports can serve as DMA targets by using a general-purpose DMA controller to access the transmit and receive buffers defined by the Ethernet buffer descriptors. Because Ethernet data buffers are located in RAM outside of the Ethernet controllers, general-purpose DMA engines can move data to or from these memory regions. Also, because Ethernet controllers automatically read buffer descriptors and send (or load) data buffers, a DMA transfer into (or out of) these buffers is effectively a transfer into (or out of) the Ethernet ports.

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 (PIC), doing so is extremely inefficient and is seldom beneficial in normal operation. The overhead of creating DMA descriptors far exceeds any savings in CPU cycles.

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/PCI-X Bus Interface

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 describes the PCI/PCI-X controller (referenced as PCI/X throughout this chapter) of this device and 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. Note that this is inconsistent with the terminology in the rest of this manual where the terms ‘word’ and ‘double word’ refer to a 32-bit and 64-bit quantity respectively. Where necessary to avoid confusion, the precise number of bits or bytes is specified.

15.1 Introduction

The PCI/X controller acts as 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.

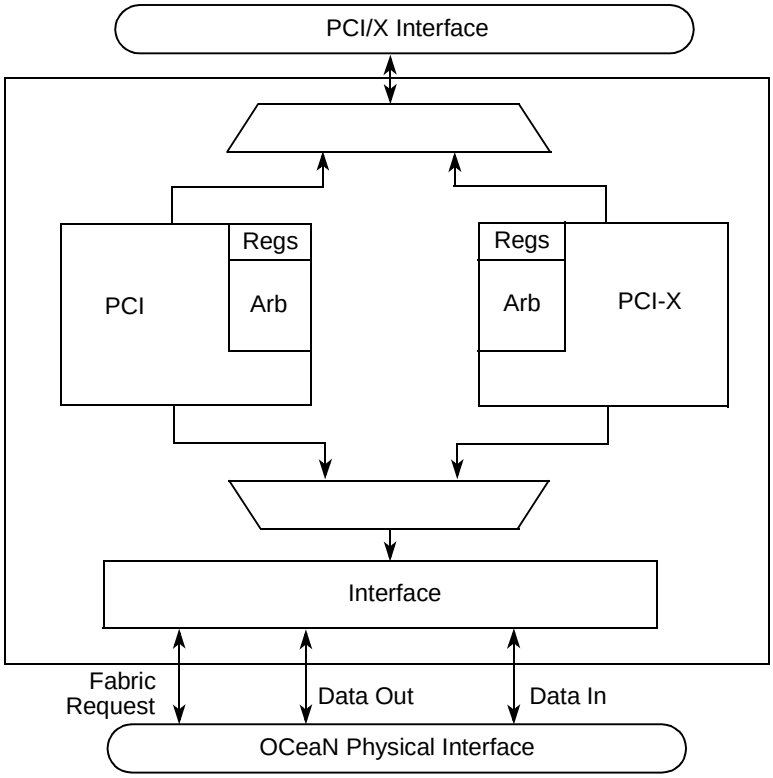


Figure 15-1. PCI/X Controller Block Diagram

15.1.1 Overview

The PCI/X controller connects the OCeaN to the 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/X interface supports address and data parity with error checking and reporting.

The PCI/X 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/X transactions simultaneously. For example, consider the case where a burst-write transaction from the MPC8560 to another PCI device

terminates with a disconnect 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/X memory space, the PCI/X I/O space, and the 256-byte PCI/X configuration space. As an initiator, the MPC8560 also supports generating PCI/X 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/X configuration registers.

The MPC8560 can function as either a PCI/X host bridge (host mode) or a peripheral device on the PCI/X bus (agent mode). See Section 15.1.3.1.1, “Host Mode,” for more information.

In agent mode, all of the PCI/X configuration registers in the MPC8560 can be programmed from the PCI/X bus. See Section 15.4.2.11.3, “PCI Configuration in Agent and Agent Lock Modes,” for more information.

The PCI/X interface provides bus arbitration for the MPC8560 and up to five other PCI/X bus masters. The arbitration algorithm is a programmable two-level, round-robin priority selector. The on-chip PCI/X arbiter can operate in both host and agent modes or it can be disabled to allow for an external PCI/X arbiter.

The MPC8560 also provides an address translation mechanism to map inbound PCI/X to OCeaN accesses and outbound OCeaN to PCI/X accesses.

15.1.1.1 MPC8560 as a PCI/X Initiator

Upon detecting an OCeaN-to-PCI/X transaction, the MPC8560 requests the use of the PCI/X bus. For OCeaN-to-PCI/X bus write operations, the MPC8560 requests mastership of the PCI/X bus when the source completes the write operation to the OCeaN. For OCeaN-to-PCI/X read operations, the MPC8560 requests mastership of the PCI/X bus when it decodes that the access is for PCI/X address space.

Once granted, the MPC8560 drives the 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/X Target

Upon detection of a PCI/X 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/X bus command, and forwards the transaction to the OCeaN control unit. On writes to local memory, data is forwarded along with the byte enables (if applicable) to the internal control unit. On reads, the data is driven on the bus and the byte enables (if applicable) 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. As a PCI/X target, the MPC8560 can issue split response and split completion transactions.

The MPC8560 supports data streaming to and from local memory. This means that data can flow between the MPC8560's PCI/X interface and local memory as long as the internal buffers are not filled.

15.1.2 Features

The following is a list of PCI features that are supported:

- PCI interface 2.2 compatible
- 66- 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
- Accesses to all PCI memory and I/O address spaces
- PCI-to-memory and memory-to-PCI streaming
- Memory prefetching of PCI read accesses
- Posting of processor-to-PCI and PCI-to-memory writes
- Selectable snoop for inbound accesses
- PCI configuration registers
- PCI 3.3-V compatible

The following is a list of PCI-X features supported:

- PCI-X revision 1.0a compatible
- Support for up to 133-MHz point-to-point connection
- Support for 32- 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 four split transactions
- Complete allowable disconnect boundary (ADB) support
- All PCI-X ordering rules enforced
- Implementation of relaxed-ordering
- PCI-X 3.3-V compatible

15.1.3 Modes of Operation

A number of parameters that affect the PCI/X controller modes of operation are determined at power-on reset (POR) by reset configuration signals as described in Chapter 4, “Reset, Clocking, and Initialization.” Table 15-1 provides a summary of these modes.

Table 15-1. POR Parameters for PCI/X Controller

Parameter	Description	Section/Page
Host/agent configuration	Selects between host and agent mode for the PCI/X and RapidIO interfaces.	4.4.3.4/4-14
PCI width selection	Selects between 32-bit or 64-bit data bus width.	4.4.3.12/4-18
PCI I/O impedance	Selects the impedance of the PCI/X I/O drivers	4.4.3.13/4-19
PCI arbiter enable	Enables the on-chip PCI/X bus arbiter	4.4.3.14/4-19
PCI debug mode enable	Selects between normal operation or debug mode for PCI_AD[62:58].	4.4.3.15/4-19
PCI-X configuration	Selects between PCI or PCI-X operation	4.4.3.16/4-20
PCI/X output hold	Selects the number of buffer delays for PCI output signals depending on whether PCI or PCI-X operation is selected. This provides flexibility in meeting minimum output hold specifications relative to SYSCLK for both PCI and PCI-X systems	4.4.3.19/4-21

15.1.3.1 Host/Agent Modes

The PCI/X controller can function as either a PCI/X host bridge (host mode) or a peripheral device on the PCI bus (agent mode). The PCI/X controller can also operate in agent configuration lock mode. Note that host/agent mode selection is determined at power-up as summarized in Section 15.5.1, “Power-On Reset Configuration Modes.”

15.1.3.1.1 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.

15.1.3.1.2 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. Note that in PCI agent mode, the PCI/X controller ignores all PCI/X memory accesses except those to the memory-mapped registers) until inbound address translation is enabled.

15.1.3.1.3 Agent Configuration Lock Mode

When the device powers up in agent configuration lock mode, it retries inbound configuration accesses until the ACL bit in the PCI bus function register is cleared. See Section 15.5.1.3, “Agent Configuration Lock Mode,” for more information.

15.1.3.2 PCI/X Bus Width (64-/32-Bit Bus)

This input configures the PCI/X interface to be in 32-bit or 64-bit extended mode of operation. The initial value is determined by the value on the $\overline{\text{PCI\_REQ64}}$ power-on reset configuration signal. See Chapter 4, “Reset, Clocking, and Initialization,” and the *MPC8560 Integrated Processor Hardware Specifications*, for more information.

15.1.3.3 PCI/X Arbiter (Internal/External Arbiter)

This input configures the on-chip PCI/X arbiter. The initial value is determined by the value on the $\overline{\text{PCI\_GNT2}}$ power-on reset configuration signal. See Chapter 4, “Reset, Clocking, and Initialization,” and the *MPC8560 Integrated Processor Hardware Specifications*.

15.1.3.4 PCI/X Bus Mode

This input configures PCI or PCI-X mode on the 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. The initial value is determined by the value on the $\overline{\text{PCI\_GNT4}}$ power-on reset configuration signal. See Chapter 4, “Reset, Clocking, and Initialization,” and the *MPC8560 Integrated Processor Hardware Specifications*.

15.1.3.5 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 bus 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 Integrated Processor Hardware Specifications*, for more information on these values and signal timing.

15.1.3.6 PCI/X Impedance

The MPC8560 has a programmable impedance for PCI/X bus signals. The initial value is determined by the value on the PCI_GNT1 power-on reset configuration signal. See Chapter 4, “Reset, Clocking, and Initialization,” and the *MPC8560 Integrated Processor Hardware Specifications*, for more information.

15.2 External Signal Descriptions

Figure 15-2 shows the external PCI/X signals.

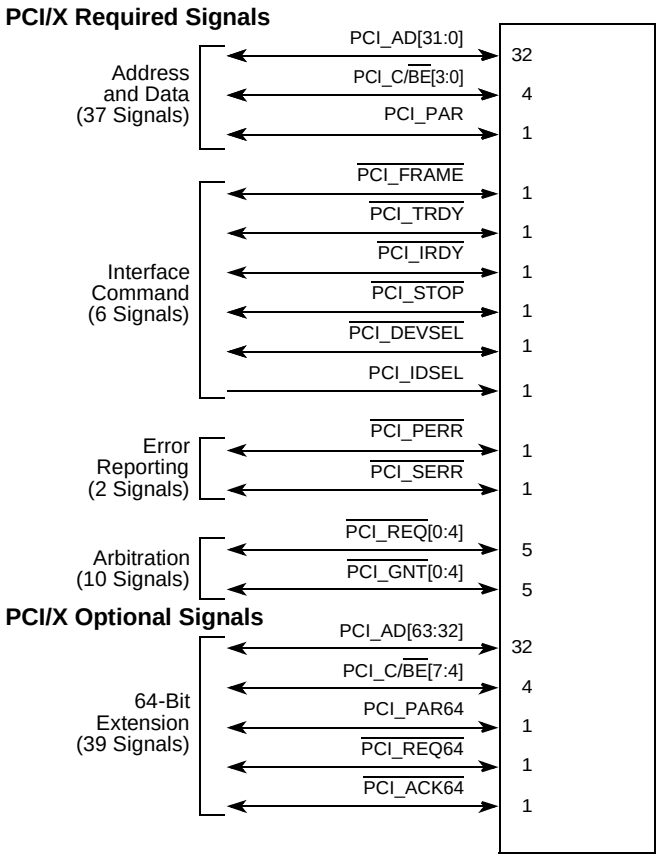


Figure 15-2. PCI/X Interface External Signals

Table 15-2 contains the detailed descriptions of the external PCI/X interface signals.

Table 15-2. PCI Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
$\overline{\text{PCI\_ACK64}}$	I/O	64-bit transaction acknowledge.Indicates that the current target supports 64-bit transfers during the data phase of the current transaction.	
	O	As an output for the bidirectional 64-bit transaction acknowledge, this signal operates 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As an input for the bidirectional 64-bit transaction acknowledge, this signal operates 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_AD[63:0]	I/O	PCI address/data bus	
	O	As outputs for the bidirectional 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 PCI_AD[63:56] define the MSB.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bidirectional 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. The PCI_AD[7:0] signals define the LSB and PCI_AD[63:56] define the MSB.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a

Table 15-2. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_C/$\overline{\text{BE}}$[7:0]	I/O	Command/byte enable. Command encodings are described in Section 15.4.2.2, "PCI Bus Commands," and Section 15.4.3.2, "PCI-X Command Encodings."	
	O	As outputs for the bidirectional 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 for PCI bus data phases. The PCI_C/$\overline{\text{BE}}$0 signal applies to the LSB.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bidirectional command/byte enable, these signals operate as described below.	
		State Meaning	Asserted/Negated—During the address phase, PCI_C/$\overline{\text{BE}}$[7:0] indicate the command that another master is sending. During the PCI bus data phase, PCI_C/$\overline{\text{BE}}$[7:0] indicate which byte lanes are valid.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
$\overline{\text{PCI\_DEVSEL}}$	I/O	Device select	
	O	As outputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a

Table 15-2. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_FRAME	I/O	Frame	
	O	As outputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bidirectional 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 Specification 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_GNT0 is an input. Note that PCI_GNT[n] is a point-to-point signal. Every master has its own bus grant signal. Also, note that these signals are also used as reset configuration signals in the MPC8560 as described in Section 4.4.3, “Power-On Reset Configuration."	
		State Meaning	Asserted—Indicates that this PCI controller granted control of the PCI bus to agent *n*. Negated—Indicates that this PCI controller did not grant control of the PCI bus to agent *n*.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_IDSEL	I	Initialization device select. 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a

Table 15-2. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_IRDY	I/O	Initiator ready	
	O	As outputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bidirectional initiator ready, these signals operate as described below.	
		State Meaning	Asserted—Indicates another PCI master can 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_PAR	I/O	PCI parity.	
	O	As outputs for the bidirectional PCI parity, these signals operate as described below.	
		State Meaning	Asserted—Indicates odd parity across PCI_AD[31:0] and PCI_C/BE[3:0] during address and data phases. Negated—Indicates even parity across PCI_AD[31:0] and PCI_C/BE[3:0] during address and data phases.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a

Table 15-2. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_PAR64	I/O	Upper DWORD parity. 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 bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
$\overline{PCI\_PERR}$	I/O	PCI parity error	
	O	As outputs for the bidirectional 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. (Driven by the PCI initiator on reads; driven by the PCI target on writes.) Negated—Indicates no error.
		Timing	Assertion/Negation—As specified by PCI Local Bus Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a **Note:** If a parity error occurs on the last data beat of a PCI-X transaction with inbound data (outbound read with split completion data or inbound write), the MPC8560 asserts $\overline{PERR}$ longer than permitted by the PCI-X specification. This condition may cause a subsequent transaction to erroneously detect a parity error. The condition may be remedied by placing a stronger pull-up resistors on the $\overline{PERR}$ signal. For example: • 1 Kohm for 133 MHz point-to-point operation • 2 Kohm for 66 MHz operation • 4 Kohm for 33 MHz operation. These values are stronger than the specification allows (5 Kohm minimum for pull-up resistors).

Table 15-2. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_REQ[4:0]	I	PCI bus request. Input signals on this PCI controller when the arbiter is enabled. When the arbiter is disabled, PCI_REQ[0] is an output. Note that PCI_REQ[n] is a point-to-point signal. Every master has its own bus request signal. Following is the state meaning for the PCI_REQ[n] input.	
		State Meaning	Asserted—Indicates that agent *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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_REQ64	I/O	64-bit transaction request. Indicates that the current master desires to transfer data using 64-bit transfers. Also used as a reset configuration signal in the MPC8560 as described in Section 4.4.3, "Power-On Reset Configuration."	
	O	As an output for the bidirectional 64-bit transaction request, this signal operates 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As an input for the bidirectional 64-bit transaction request, this signal operates 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_SERR	I/O	PCI system error	
	O	As outputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a

Table 15-2. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_STOP	I/O	Stop.	
	O	As outputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
PCI_TRDY	I/O	Target ready.	
	O	As outputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a
	I	As inputs for the bidirectional 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 Specification Rev 2.2 or by PCI-X Addendum Rev 1.0a

15.3 Memory Map/Register Definitions

The PCI/X interface unit of the MPC8560 supports the following register types:

- 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/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/X Configuration Header," and its subsections.

15.3.1 PCI/X Memory Mapped Registers

The 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/X side) plus the offset of the specific register to be accessed. Note that all memory-mapped registers (except the PCI/X configuration data register, PCI CFG_DATA) must only be accessed as 32-bit quantities.

Table 15-3 lists the memory-mapped registers.

Table 15-3. PCI/X Memory-Mapped Register Map

Offset	Register	Access	Reset	Section/Page
PCI/X Configuration Access Registers				
0x0_8000	CFG_ADDR—PCI/X configuration address	R/W	0x0000_0000	15.3.1.1.1/15-18
0x0_8004	CFG_DATA—PCI/X configuration data	R/W	0x0000_0000	15.3.1.1.1/15-18
0x0_8008	INT_ACK—PCI/X interrupt acknowledge	R	0x0000_0000	15.3.1.1.3/15-20
0x0_800C–0x0_8BFC	Reserved	—	—	—
PCI/X ATMU Registers—Outbound and Inbound				
0x0_8C00–0x0_8C3C–Outbound Window 0 (default)				
0x0_8C00	POTAR0—PCI/X outbound window 0 (default) translation address register	R/W	0x0000_0000	15.3.1.2.1/15-21
0x0_8C04	POTEAR0—PCI/X outbound window 0 (default) translation extended address register	R/W	0x0000_0000	15.3.1.2.2/15-21
0x0_8C08	Reserved	—	—	
0x0_8C0C	Reserved	—	—	
0x0_8C10	POWAR0—PCI/X outbound window 0 (default) attributes register	R/W	0x8004_401F	15.3.1.2.4/15-22
0x0_8C14–0x0_8C1C	Reserved	—	—	

Table 15-3. PCI/X Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8C20–0x0_8C3C—Outbound Window 1				
0x0_8C20	POTAR1—PCI/X outbound window 1 translation address register	R/W	0x0000_0000	15.3.1.2.1/15-21
0x0_8C24	POTEAR1—PCI/X outbound window 1 translation extended address register	R/W	0x0000_0000	15.3.1.2.2/15-21
0x0_8C28	POWBAR1—PCI/X outbound window 1 base address register	R/W	0x0000_0000	15.3.1.2.3/15-22
0x0_8C2C	Reserved	—	—	
0x0_8C30	POWAR1—PCI/X outbound window 1 attributes register	R/W	0x0000_0000	15.3.1.2.4/15-22
0x0_8C34–0x0_8C3C	Reserved	—	—	
0x0_8C40–0x0_8C5C—Outbound Window 2				
0x0_8C40	POTAR2—PCI/X outbound window 2 translation address register	R/W	0x0000_0000	15.3.1.2.1/15-21
0x0_8C44	POTEAR2—PCI/X outbound window 2 translation extended address register	R/W	0x0000_0000	15.3.1.2.2/15-21
0x0_8C48	POWBAR2—PCI/X outbound window 2 base address register	R/W	0x0000_0000	15.3.1.2.3/15-22
0x0_8C4C	Reserved	—	—	
0x0_8C50	POWAR2—PCI/X outbound window 2 attributes register	R/W	0x0000_0000	15.3.1.2.4/15-22
0x0_8C54–0x0_8C5C	Reserved	—	—	
0x0_8C60–0x0_8C7C—Outbound Window 3				
0x0_8C60	POTAR3—PCI/X outbound window 3 translation address register	R/W	0x0000_0000	15.3.1.2.1/15-21
0x0_8C64	POTEAR3—PCI/X outbound window 3 translation extended address register	R/W	0x0000_0000	15.3.1.2.2/15-21
0x0_8C68	POWBAR3—PCI/X outbound window 3 base address register	R/W	0x0000_0000	15.3.1.2.3/15-22
0x0_8C6C	Reserved	—	—	
0x0_8C70	POWAR3—PCI/X outbound window 3 attributes register	R/W	0x0000_0000	15.3.1.2.4/15-22
0x0_8C74–0x0_8C7C	Reserved	—	—	
0x0_8C80–0x0_8C9C—Outbound Window 4				
0x0_8C80	POTAR4—PCI/X outbound window 4 translation address register	R/W	0x0000_0000	15.3.1.2.1/15-21
0x0_8C84	POTEAR4—PCI/X outbound window 4 translation extended address register	R/W	0x0000_0000	15.3.1.2.2/15-21

Table 15-3. PCI/X Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8C88	POWBAR4—PCI/X outbound window 4 base address register	R/W	0x0000_0000	15.3.1.2.3/15-22
0x0_8C8C	Reserved	—	—	
0x0_8C90	POWAR4—PCI/X outbound window 4 attributes register	R/W	0x0000_0000	15.3.1.2.4/15-22
0x0_8C94–0x0_8D9C	Reserved	—	—	
0x0_8DA0–0x0_8DBC–Inbound Window 3				
0x0_8DA0	PITAR3—PCI/X inbound window 3 translation address register	R/W	0x0000_0000	15.3.1.3.1/15-25
0x0_8DA4	Reserved	—	—	
0x0_8DA8	PIWBAR3—PCI/X inbound window 3 base address register	R/W	0x0000_0000	15.3.1.3.2/15-25
0x0_8DAC	PIWBEAR3—PCI/X inbound window 3 base extended address register	R/W	0x0000_0000	15.3.1.3.3/15-26
0x0_8DB0	PIWAR3—PCI/X inbound window 3 attributes register	R/W	0x0000_0000	15.3.1.3.4/15-26
0x0_8DB4–0x0_8DBC	Reserved	—	—	
0x0_8DC0–0x0_8DDC–Inbound Window 2				
0x0_8DC0	PITAR2—PCI/X inbound window 2 translation address register	R/W	0x0000_0000	15.3.1.3.1/15-25
0x0_8DC4	Reserved	—	—	
0x0_8DC8	PIWBAR2—PCI/X inbound window 2 base address register	R/W	0x0000_0000	15.3.1.3.2/15-25
0x0_8DCC	PIWBEAR2—PCI/X inbound window 2 base extended address register	R/W	0x0000_0000	15.3.1.3.3/15-26
0x0_8DD0	PIWAR2—PCI/X inbound window 2 attributes register	R/W	0x0000_0000	15.3.1.3.4/15-26
0x0_8DD4–0x0_8DDC	Reserved	—	—	
0x0_8DE0–0x0_8DFC–Inbound Window 1				
0x0_8DE0	PITAR1—PCI/X inbound window 1 translation address register	R/W	0x0000_0000	15.3.1.3.1/15-25
0x0_8DE4	Reserved	—	—	
0x0_8DE8	PIWBAR1—PCI/X inbound window 1 base address register	R/W	0x0000_0000	15.3.1.3.2/15-25
0x0_8DEC	Reserved	—	—	
0x0_8DF0	PIWAR1—PCI/X inbound window 1 attributes register	R/W	0x0000_0000	15.3.1.3.4/15-26
0x0_8DF4–0x0_8DFC	Reserved	—	—	

Table 15-3. PCI/X Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
PCI/X Error Management Registers				
0x0_8E00	ERR_DR—PCI/X error detect register	Special	0x0000_0000	15.3.1.4.1/15-29
0x0_8E04	ERR_CAP_DR—PCI/X error capture disabled register	R/W	0x0000_0000	15.3.1.4.2/15-30
0x0_8E08	ERR_EN—PCI/X error enable register	R/W	0x0000_0000	15.3.1.4.3/15-31
0x0_8E0C	ERR_ATTRIB—PCI/X error attributes capture register	R/W	0x0000_0000	15.3.1.4.4/15-32
0x0_8E10	ERR_ADDR—PCI/X error address capture register	R/W	0x0000_0000	15.3.1.4.5/15-33
0x0_8E14	ERR_EXT_ADDR—PCI/X error extended address capture register	R/W	0x0000_0000	15.3.1.4.6/15-34
0x0_8E18	ERR_DL—PCI/X error data low capture register	R/W	0x0000_0000	15.3.1.4.7/15-34
0x0_8E1C	ERR_DH—PCI/X error data high capture register	R/W	0x0000_0000	15.3.1.4.8/15-34
0x0_8E20	GAS_TIMR—PCI/X gasket timer register	R/W	0x0000_0000	15.3.1.4.9/15-35
0x0_8E24	PCIX_TIMR—PCI-X split completion timer register	R/W	0x0000_0000	15.3.1.4.10/15-36
0x0_8E28–0x0_8EFC	Reserved	—	—	
0x0_8F00–0x0_8FFC	Reserved for debug	—	—	

15.3.1.1 PCI/X Configuration Access Registers

The PCI/X configuration header, shown in Figure 15-25 and Figure 15-26, is accessed through an indirect method using a pair of 32-bit memory-mapped access registers, CFG_ADDR at offset 0x0_8000 and CFG_DATA at offset 0x0_8004.

15.3.1.1.1 PCI/X Configuration Address Register (CFG_ADDR)

The CFG_ADDR register is shown in Figure 15-3.

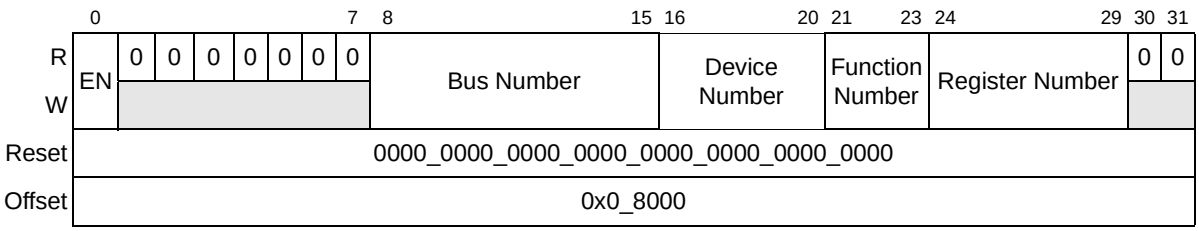


Figure 15-3. PCI/X CFG_ADDR Register

Table 15-4 describes the bit settings for the CFG_ADDR register.

Table 15-4. PCI/X CFG_ADDR Field Descriptions

Bits	Name	Description
0	Enable	Allow a PCI configuration access when PCI 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 header of the MPC8560 itself.

See Section 15.4.2.11.2, “Accessing the PCI Configuration Space in Host Mode,” and Section 15.4.2.11.3, “PCI Configuration in Agent and Agent Lock Modes,” for usage of PCI/X CFG_ADDR.

15.3.1.1.2 PCI/X Configuration Data Register (CFG_DATA)

The CFG_DATA register is shown in Figure 15-3.

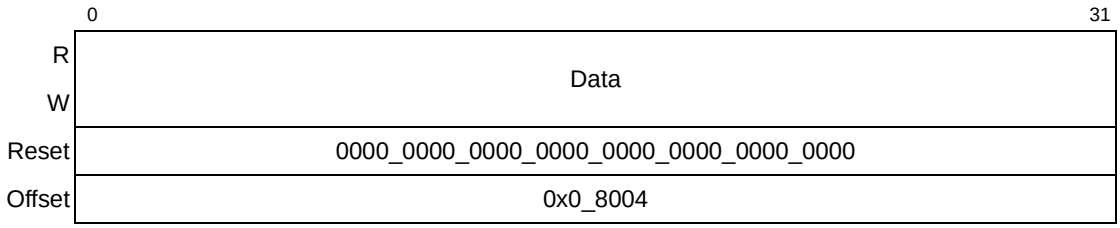


Figure 15-4. PCI/X CFG_DATA Register

Table 15-5 describes the bit settings for the CFG_DATA register

Table 15-5. PCI/X CFG_DATA Field Descriptions

Bits	Name	Description
0–31	Data	A read or write to this register starts a PCI configuration cycle if the PCI CFG_ADDR enable bit is set. If the enable bit is not set, a PCI I/O transaction is generated.

The CFG_DATA register is a 4-byte window into the little-endian PCI/X configuration header data structure; therefore byte addressing within the CFG_DATA register uses little-endian convention. Note that CFG_DATA may contain 1, 2, 3, or 4 bytes depending on the size of the register being accessed.

See Section 15.4.2.11.2, “Accessing the PCI Configuration Space in Host Mode,” and Section 15.4.2.11.3, “PCI Configuration in Agent and Agent Lock Modes,” for use of CFG_DATA.

15.3.1.1.3 PCI/X Interrupt Acknowledge Register (INT_ACK)

An external PCI/X interrupt acknowledge transaction is generated by reading the INT_ACK register at offset 0x0_8008. PCI INT_ACK is read-only and a write to that address results in nothing. The INT_ACK register is shown in Figure 15-5.

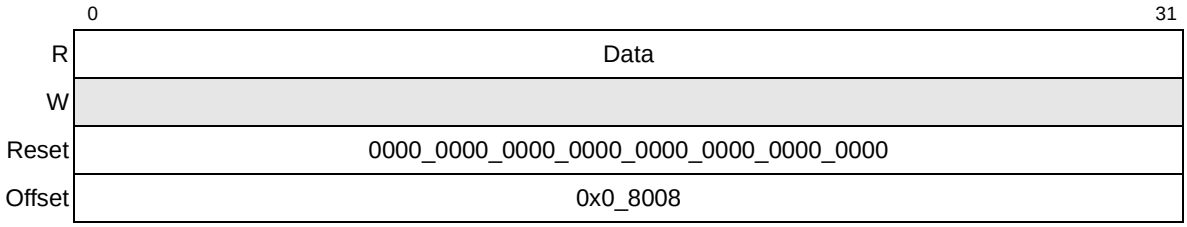


Figure 15-5. PCI/X INT_ACK Register

Table 15-6 describes the bit settings for the INT_ACK register.

Table 15-6. PCI/X INT_ACK Field Descriptions

Bits	Name	Description
0–31	Data	A read to this register generates a PCI interrupt acknowledge cycle.

15.3.1.2 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 based on the granularity specified by the window size. If two outbound ATMU windows overlap in the local address space, the mapping of the lower numbered window has precedence over the higher numbered window. Note that outbound translation windows must not overlap the configuration access registers.

Window 0 is the default window and is the only window enabled upon reset. The default outbound register set is used when a transaction misses in all other outbound windows.

15.3.1.2.1 PCI/X Outbound Translation Address Registers (POTAR*n*)

The PCI/X outbound translation address registers (POTAR*n*) select the starting addresses in the PCI address space for hits in the PCI/X outbound windows. The translated address is created by concatenating the transaction offset to this translation address. The format of the POTAR*n* is shown in Figure 15-6.

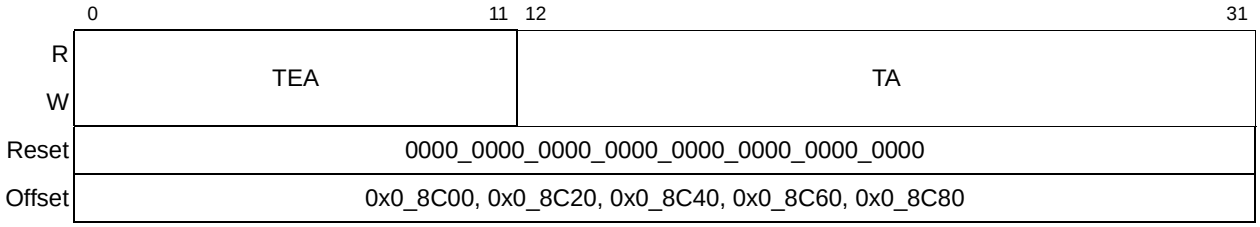


Figure 15-6. PCI/X Outbound Translation Address Registers

Table 15-7 describes the fields of the POTAR*n* registers.

Table 15-7. POTAR*n* Field Descriptions

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 specified in POWAR*n*[OWS], the low-order bits of this field may be ignored.

15.3.1.2.2 PCI/X Outbound Translation Extended Address Registers (POTEAR*n*)

The PCI/X outbound translation extended address registers (POTEAR*n*), shown in Figure 15-7, contain the most significant bits of a 64-bit translation address.

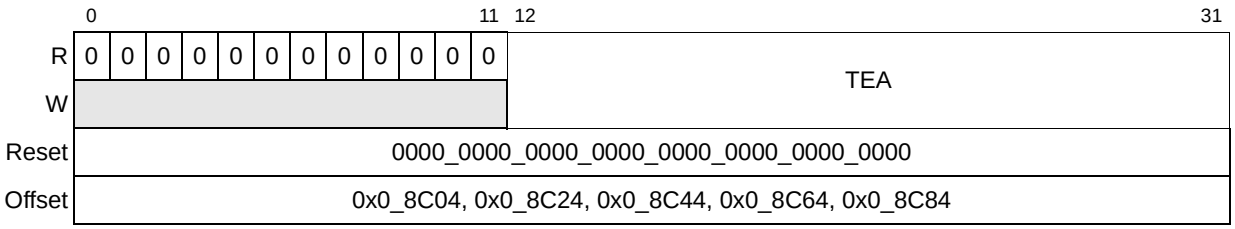


Figure 15-7. PCI/X Outbound Translation Extended Address Registers

Table 15-8 describes the fields of the POTEAR*n*.

Table 15-8. POTEAR*n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	TEA	Translation extended address. Represents bits [63–44] of a 64-bit PCI address (used in conjunction with POTAR*n*)

15.3.1.2.3 PCI/X Outbound Window Base Address Registers (POWBAR*n*)

The PCI/X outbound window base address registers (POWBAR*n*), shown in Figure 15-8, 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 enabled and used when the other windows miss.

Note that POWBAR0 (for outbound ATMU window 0) is not used, because window 0 is the default window used when no other windows match. POWBAR0 may be read from and written to, but the value is ignored.

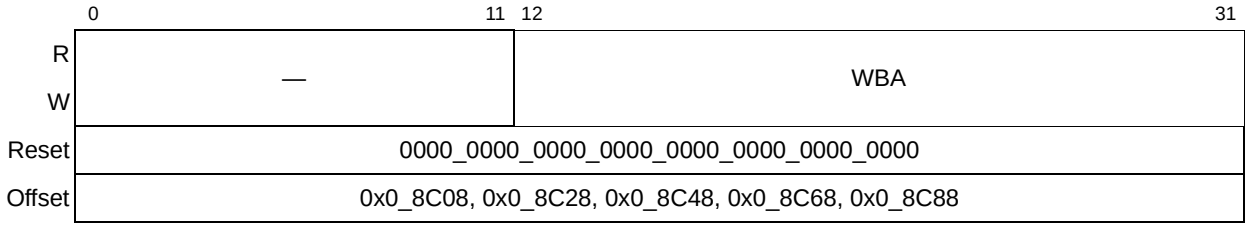


Figure 15-8. PCI/X Outbound Window Base Address Registers

Table 15-9 describes the field of the POWBAR*n*.

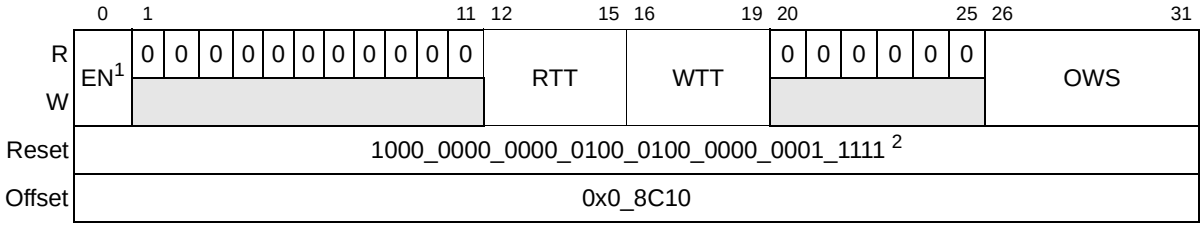
Table 15-9. POWBAR*n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved, should be cleared.
12–31	WBA	Window base address. 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.

15.3.1.2.4 PCI/X Outbound Window Attributes Registers (POWAR*n*)

The PCI/X outbound window attributes registers (POWAR*n*) 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.

The default window attribute register, POWAR0, is shown in Figure 15-9. Note that the fields for all of the POWAR*n* registers are the same, only the reset values are different.



[1] For POWAR0, translation is always enabled. The enable field (EN) may be read and written, but the value is ignored.

[2] The default window is enabled, configured for memory read and memory write, and set to an OWS size of 4 Gbytes.

Figure 15-9. PCI/X Outbound Window Attributes Register 0 (Default)

POWAR1–POWAR4 are shown in Figure 15-10.

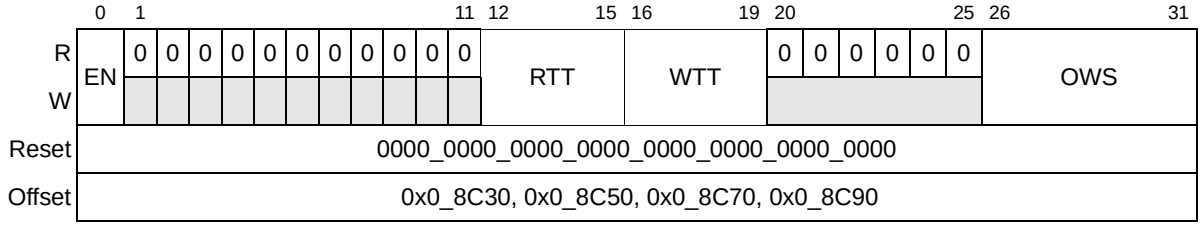


Figure 15-10. PCI/X Outbound Window Attributes Registers 1–4

Table 15-10 describes the fields for the POWAR*n* registers.

Table 15-10. POWAR*n* Field Descriptions

Bits	Name	Description
0	EN	Enable. Enables this address translation. Note that for POWAR0, translation is always enabled. The enable field (EN) may be read and written, but the value is ignored.
1–11	—	Reserved
12–15	RTT	Read transaction type to run on PCI 0000–0011 Reserved 0100 Memory Read 0101–0111 Reserved 1000 I/O Read 1001–1111 Reserved
16–19	WTT	Writetransaction type to run on PCI 0000–0011 Reserved 0100 Memory Write 0101–0111 Reserved 1000 I/O Write 1001–1111 Reserved

Table 15-10. POWAR*n* Field Descriptions (continued)

Bits	Name	Description
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 4 Kbytes. 000000–001011 4 Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000–111111 Reserved The default POWAR register (0x0_8C10) has an OWS value of 011111. Also note that for POWAR0, setting OWS to less than 4 Gbytes causes addresses that miss in the other outbound windows to be aliased to the smaller address range defined by POWAR0[OWS] and POTAR0.

15.3.1.3 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 is comprised of four windows—a configuration window and three general translation windows. The configuration window has higher priority than all other inbound ATMU windows and takes precedence over them if there is an overlap.

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 15-25. 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's base address and translation address 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 numbered window are applied; however, it is illegal for an inbound window to overlap the PCSRBAR window. In addition, if inbound ATMU windows are overlapped, the ATMU windows must not map to the same address with different sets of attributes.

Note that PCSRBAR 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 PCI CFG_ADDR and PCI CFG_DATA registers. See Section 15.3.1.1.1, "PCI/X Configuration Address Register (CFG_ADDR)," Section 15.3.1.1.2, "PCI/X Configuration Data Register (CFG_DATA)," and Section 15.3.2.11, "PCI Base Address

Registers.” All accesses to PCSRBAR have an automatic internal byte lane redirection from the little-endian PCI bus to the big-endian CCSRBAR configuration space.

15.3.1.3.1 PCI/X Inbound Translation Address Registers (PITAR*n*)

The PCI/X inbound translation address registers (PITAR*n*) points to the beginning of the local address space for the inbound window. The translated address is created by concatenating the transaction offset to this translation address. The format of the PITAR*n* is shown in Figure 15-11.

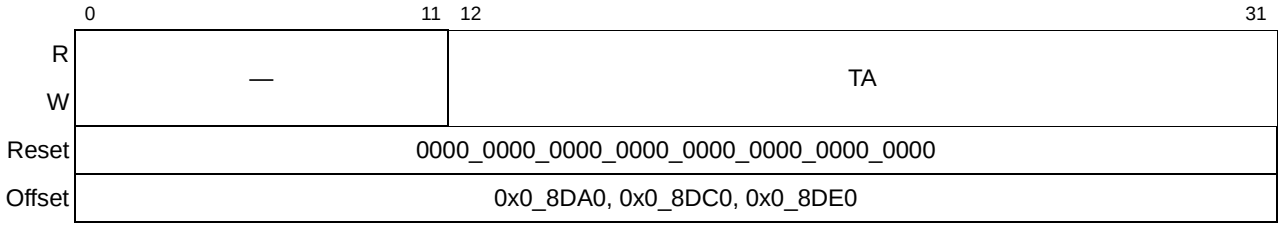


Figure 15-11. PCI/X Inbound Translation Address Registers

Table 15-11 describes the fields of the PITAR*n* registers.

Table 15-11. PITAR*n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved, should be cleared.
12–31	TA	Translation 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.

15.3.1.3.2 PCI/X Inbound Window Base Address Registers (PIWBAR*n*)

PCI/X inbound window base address registers (PIWBAR*n*), shown in Figure 15-12, select the PCI/X base address for windows translated to the MPC8560 local address space. Inbound transaction addresses are compared to these windows. If a PCI/X transaction does not fall in one of these spaces, the PCI/X interface does not assert DEVSEL.

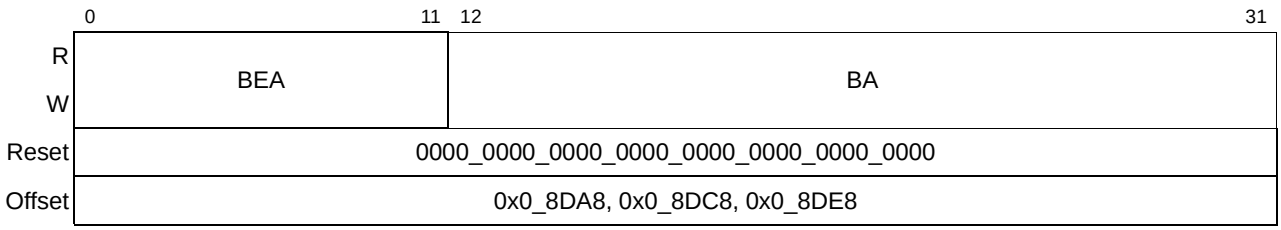


Figure 15-12. PCI/X Inbound Window Base Address Registers

Table 15-12 describes the fields of the PIWBAR*n* registers.

Table 15-12. PIWBAR*n* Field Descriptions

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.

15.3.1.3.3 PCI/X Inbound Window Base Extended Address Registers (PIWBEAR*n*)

The PCI/X inbound window base extended address registers (PIWBEAR*n*), shown in Figure 15-13, contain the msbs 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.

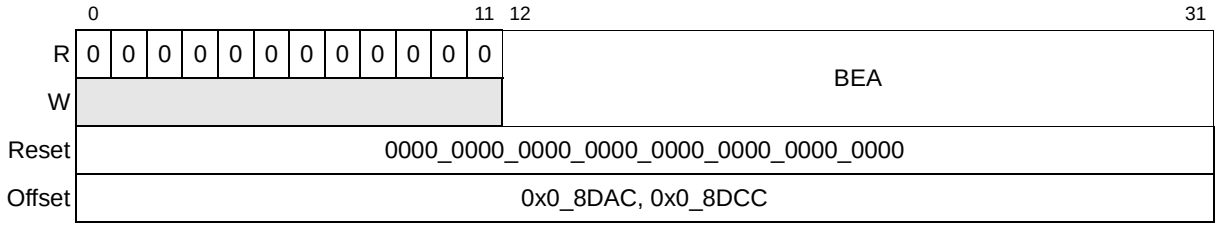


Figure 15-13. PCI/X Inbound Window Base Extended Address Registers

Table 15-13 describes the fields of the PIWBEAR*n* registers.

Table 15-13. PIWBEAR*n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	BEA	Base extended address. Corresponds to bits 63–44 of a 64-bit PCI base address.

15.3.1.3.4 PCI/X Inbound Window Attributes Registers (PIWAR*n*)

The PCI/X inbound window attributes registers (PIWAR*n*) define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed. The format of the PIWBAR*n* is shown in Figure 15-14.

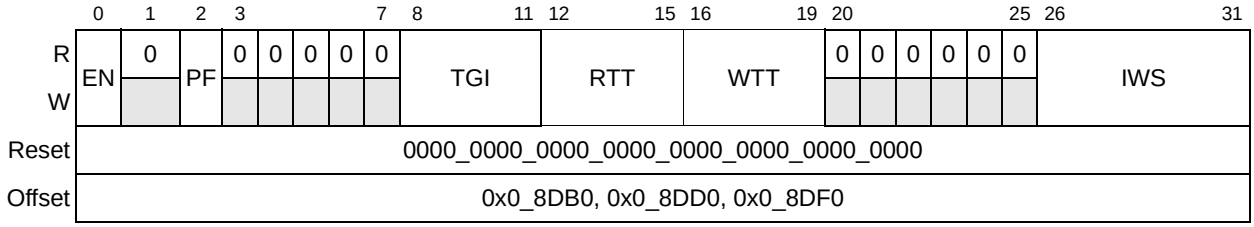


Figure 15-14. PCI/X Inbound Window Attributes Registers

Table 15-14 describes the fields of the PIWAR*n* registers.

Table 15-14. PIWAR*n* Field Descriptions

Bits	Name	Description
0	EN	Enable. Enables this address translation
1	—	Reserved
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

Table 15-14. PIWAR*n* Field Descriptions (continued)

Bits	Name	Description
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
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 is 4 Kbytes. 000000–001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000–111111 Reserved For configuration and run-time registers, the window size is fixed at 010011 1-Mbyte window size For register set 0, the window size is limited to 4 Gbytes or smaller.

15.3.1.4 PCI/X Error Management Registers

When a PCI/X error is detected, the appropriate error bit is set in the PCI/X error detect register. Subsequent errors set the appropriate error bits in the error detection registers, but relevant information (attributes, address, and data) is captured only for the first error. The PCI/X error detect register is a write-1-to-clear type register. That is, reading from this register occurs normally; however, write operations are different in that the 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 25 and not affect any other bits in the register, the value 0x0000_0040 is written to the register.

The error bit is set regardless of the state of the corresponding error enable bit in the PCI/X error enable register. The error 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.

Note that some errors are reported in two bits—one in the PCI/X error detect register (ERR_DR) and another in the PCI bus status register in the PCI/X configuration header. These bits must be cleared separately; that is, clearing one does not clear the other. For example, clearing the

ERR_DR[Mstr abort error] does not clear the received master abort bit in the PCI bus status register. In these cases, both bits must be cleared before further error reporting can occur. Refer to Table 15-57 for PCI mode error actions and Table 15-65 for PCI-X mode error actions. Likewise, some errors are enabled by programming two bits—one in the PCI/X error enable register and another in the PCI bus command register in the PCI/X configuration header.

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_FFFF.

If a data parity error occurs during an inbound configuration write access, the error is reported and captured. However, the erroneous data is written to the register specified in the transaction. Therefore, PCI data parity error recovery routines must include reinitialization of the PCI/X configuration register if the error occurred during a configuration write.

A dual address cycle (DAC) transaction on the PCI bus has two address phases. The command of the first phase is DAC and the command of the second phase is the actual transaction type. If an error occurs on a DAC transaction, the DAC command is stored in command field in the PCI/X error attribute capture register.

In PCI-X mode, the MPC8560 does not check the parity on the data of a split response to an outbound read transaction because no data is actually transferred. However, outbound writes that end with a split response do transfer data and thus parity is checked for these transactions. Also note that the MPC8560 does drive the correct parity for inbound read transactions that it terminates with a split response.

See Section 15.4.2.13, “PCI Error Functions,” and Section 15.4.3.8, “PCI-X Error Functions,” for more details on error handling.

15.3.1.4.1 PCI/X Error Detect Register (ERR_DR)

	0	1–20	21	22	23	24	25	26	27	28	29	30	31
R W	Multiple PCI Errors		Addr Parity error	Rcvd SERR error	Mstr PERR error	Trgt PERR error	Mstr abort error	Trgt abort error	OWMSV error	ORMSV error	IRMSV error	SCM error	TOE error
Reset	0000_0000_0000_0000_0000_0000_0000_0000												
Offset	0x0_8E00												

Figure 15-15. PCI/X Error Detect Register (ERR_DR)

Table 15-15 describes ERR_DR fields. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and an error occurs, the appropriate parity detect and master-abort bits in ERR_DR must be cleared and the appropriate enable bits in ERR_EN must be

set to ensure that an interrupt is generated. See Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

Table 15-15. ERR_DR Field Descriptions

Bits	Name	Description
0	Multiple PCI errors	0 Multiple PCI errors of the same type were not detected (write-1-to-clear) 1 Multiple PCI errors of the same type were detected
1–20	—	Reserved
21	Addr Parity error	Address parity error (write-1-to-clear)
22	Rcvd $\overline{\text{SERR}}$ error	Received $\overline{\text{SERR}}$ error (write-1-to-clear)
23	Mstr $\overline{\text{PERR}}$ error	Master $\overline{\text{PERR}}$ error (write-1-to-clear)
24	Trgt $\overline{\text{PERR}}$ error	Target $\overline{\text{PERR}}$ error (write-1-to-clear)
25	Mstr abort error	Master-abort error (write-1-to-clear)
26	Trgt abort error	Target abort error (write-1-to-clear)
27	OWMSV error	Outbound write memory space violation error (write-1-to-clear)
28	ORMSV error	Outbound read memory space violation error (write-1-to-clear)
29	IRMSV error	PCI/X inbound read memory space violation error (write-1-to-clear)
30	SCM error	PCI/X split completion message error (write-1-to-clear)
31	TOE error	PCI/X time out error (write-1-to-clear)

15.3.1.4.2 PCI/X Error Capture Disable Register (ERR_CAP_DR)

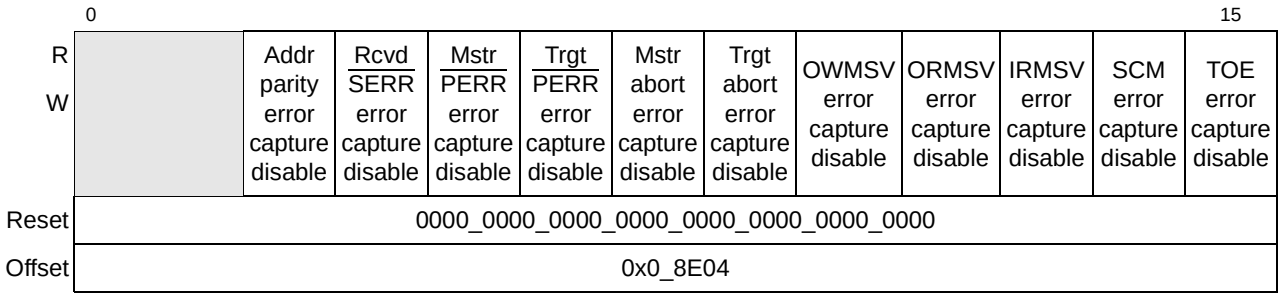


Figure 15-16. PCI/X Error Capture Disable Register (ERR_CAP_DR)

Table 15-16. ERR_CAP_DR Field Descriptions

Bits	Name	Description
0–20	—	Reserved
21	Addr parity error capture disable	Disable capture for address parity errors
22	Rcvd $\overline{\text{SERR}}$ error capture disable	Disable capture for received $\overline{\text{SERR}}$ errors
23	Mstr $\overline{\text{PERR}}$ error capture disable	Disable capture for master $\overline{\text{PERR}}$ errors
24	Trgt $\overline{\text{PERR}}$ error capture disable	Disable capture for target $\overline{\text{PERR}}$ errors
25	Mstr abort error capture disable	Disable capture for master-abort errors
26	Trgt abort error capture disable	Disable capture for target abort errors
27	OWMSV error capture disable	Disable capture for outbound write memory space violation errors
28	ORMSV error capture disable	Disable capture for outbound read memory space violation errors
29	IRMSV error capture disable	Disable capture for PCI/X inbound read memory space violation errors
30	SCM error capture disable	Disable capture for PCI/X split completion message errors
31	TOE error capture disable	Disable capture for PCI/X time out errors

15.3.1.4.3 PCI/X Error Enable Register (ERR_EN)

	0–20	21	22	23	24	25	26	27	28	29	30	31
R W		Addr parity error enable	Rcvd $\overline{\text{SERR}}$ error enable	Mstr $\overline{\text{PERR}}$ error enable	Trgt $\overline{\text{PERR}}$ error enable	Mstr abort error enable	Trgt abort error enable	OWMSV error enable	ORMSV error enable	IRMSV error enable	SCM error enable	TOE error enable
Reset	0000_0000_0000_0000_0000_0000_0000_0000											
Offset	0x0_8E08											

Figure 15-17. PCI/X Error Enable Register (ERR_EN)

Table 15-17 describes ERR_DR fields. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt unless it is disabled by clearing HID1[RFXE]. If RFXE is zero and this error occurs, the appropriate parity detect and master-abort bits in ERR_DR must be cleared and the appropriate ERR_EN bits must be set to ensure that an interrupt is generated. See Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

Table 15-17. ERR_EN Field Descriptions

Bits	Name	Description
0–20	—	Reserved
21	Addr parity error enable	Enable reporting address parity errors
22	Rcvd $\overline{\text{SERR}}$ error enable	Enable reporting received $\overline{\text{SERR}}$ errors
23	Mstr $\overline{\text{PERR}}$ error enable	Enable reporting master $\overline{\text{PERR}}$ errors
24	Trgt $\overline{\text{PERR}}$ error enable	Enable reporting target $\overline{\text{PERR}}$ errors
25	Mstr abort error enable	Enable reporting master-abort errors
26	Trgt abort error enable	Enable reporting target abort errors
27	OWMSV error enable	Enable reporting outbound write memory space violation errors
28	ORMSV error enable	Enable reporting outbound read memory space violation errors
29	IRMSV error enable	Enable reporting PCI/X inbound read memory space violation errors
30	SCM error enable	Enable reporting PCI/X split completion message errors
31	TOE error enable	Enable reporting PCI/X time out errors

15.3.1.4.4 PCI/X Error Attributes Capture Register (ERR_ATTRIB)

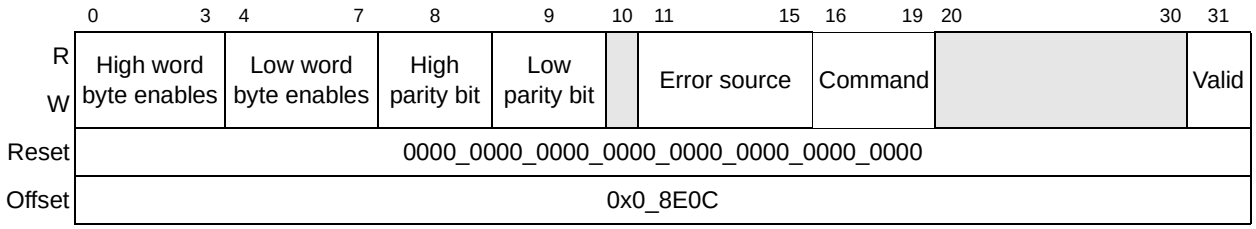


Figure 15-18. PCI/X Error Attributes Capture Register (ERR_ATTRIB)

Table 15-18. ERR_ATTRIB Field Descriptions

Bits	Name	Description
0–3	High word byte enables	PCI byte enables for most significant word of the double word
4–7	Low word byte enables	PCI byte enables for least significant word of the double word
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

Table 15-18. ERR_ATTRIB Field Descriptions (continued)

Bits	Name	Description
11–15	Error source	The source of the PCI transaction 00000 PCI/X interface 00001 Reserved 00010–00011 Reserved 00100 Local bus controller 00101–01000 Reserved 01001 Reserved 01010–01011 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 TSEC1 11001 TSEC2 11010 reserved 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

15.3.1.4.5 PCI/X Error Address Capture Register (ERR_ADDR)

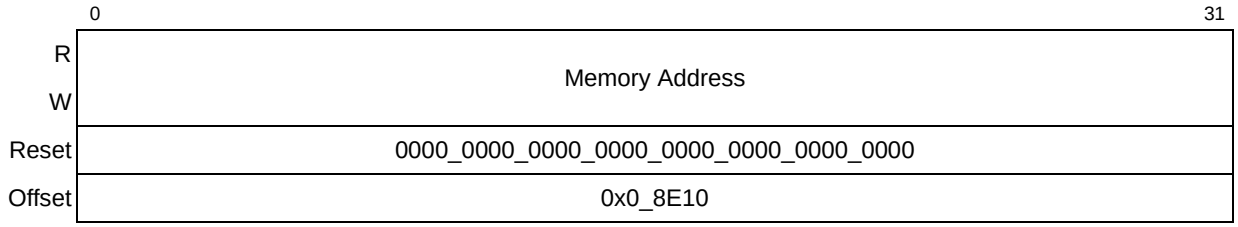


Figure 15-19. PCI/X Error Address Capture Register (ERR_ADDR)

Table 15-19. ERR_ADDR Field Descriptions

Bits	Name	Description
0–31	Memory address	Memory transaction address

15.3.1.4.6 PCI/X Error Extended Address Capture Register (ERR_EXT_ADDR)

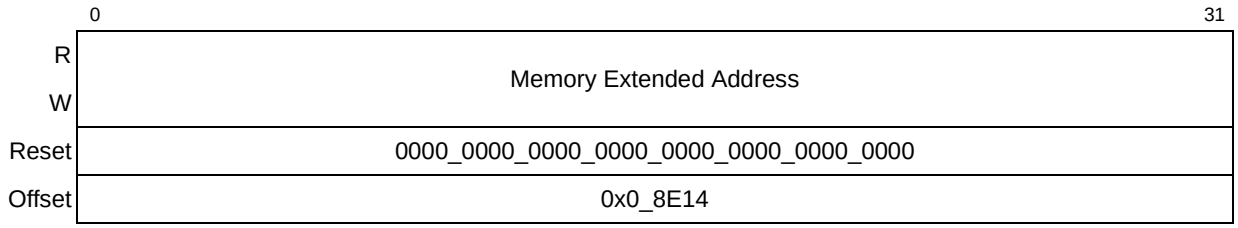


Figure 15-20. PCI/X Error Extended Address Capture Register (ERR_EXT_ADDR)

Table 15-20. ERR_EXT_ADDR Field Descriptions

Bits	Name	Description
0–31	Memory extended address	Memory transaction extended address

15.3.1.4.7 PCI/X Error Data Low Capture Register (ERR_DL)

Note that for inbound reads that have data parity errors, only the address (ERR_ADDR and ERR_EXT_ADDR) and attributes (ERR_ATTRIB) are captured. The data is not captured. Also note that PCI-X split completion error messages for outbound DWORD writes are incorrectly reported in the PCI/X error data low capture register (ERR_DL).

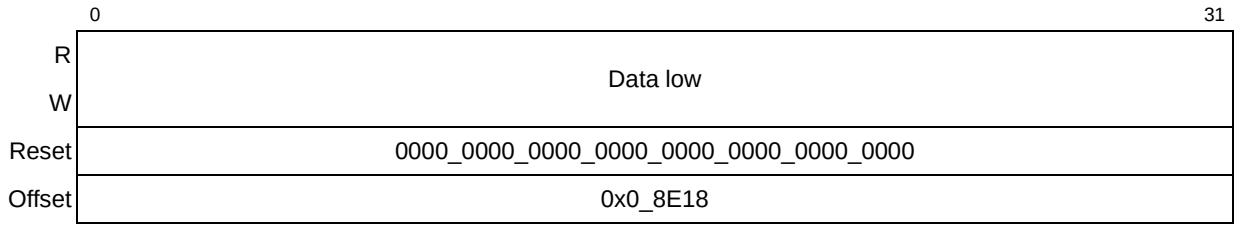


Figure 15-21. PCI/X Error Data Low Capture Register (ERR_DL)

Table 15-21. ERR_DL Field Description

Bits	Name	Description
0–31	Data low	Least significant PCI bus data word

15.3.1.4.8 PCI/X Error Data High Capture Register (ERR_DH)

Note that for inbound reads that have data parity errors, only the address (ERR_ADDR and ERR_EXT_ADDR) and attributes (ERR_ATTRIB) are captured. The data is not captured.

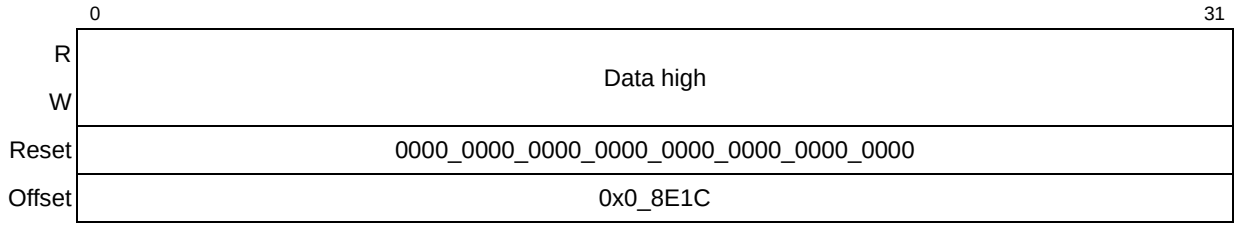


Figure 15-22. PCI/X Error Data High Capture Register (ERR_DH)

Table 15-22. ERR_DH Field Description

Bits	Name	Description
0–31	Data high	Most significant PCI bus data word (only valid with 64-bit PCI bus)

15.3.1.4.9 PCI/X Gasket Timer Register (GAS_TIMR)

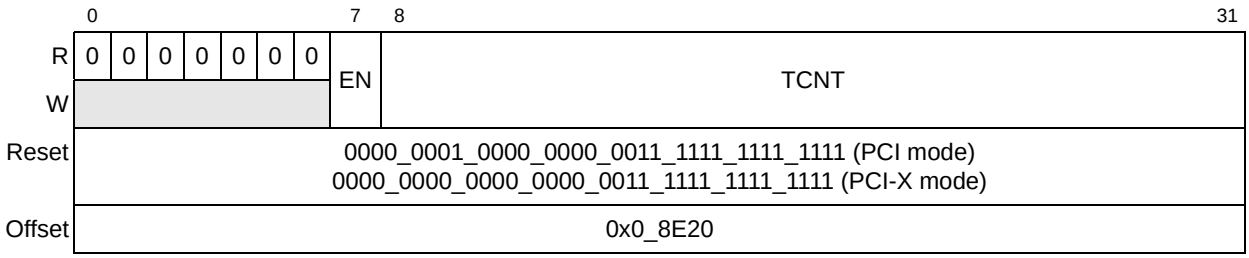


Figure 15-23. PCI/X Gasket Timer Register (GAS_TIMR)

Table 15-23. GAS_TIMR Field Descriptions

Bits	Name	Description
0–6	—	Reserved
7	EN	Gasket timer enable. 0 PCI-X default: Gasket timer is disabled. 1 PCI default: Gasket timer is enabled.
8–24	TCNT	Number of system clocks to purge a non-prefetchable inbound read buffer

15.3.1.4.10 PCI-X Split Completion Timer Register (PCIX_TIMR)

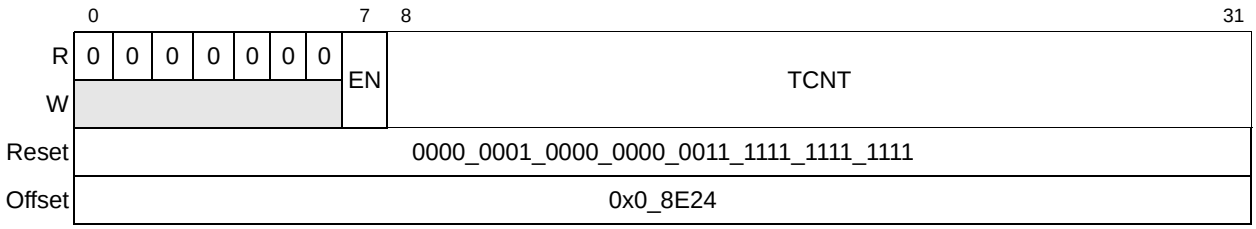


Figure 15-24. PCI-X Split Completion Timer Register (PCIX_TIMR)

Table 15-24. PCIX_TIMR Field Descriptions

Bits	Name	Description
0–6	—	Reserved
7	EN	PCI-X timer enable 0 PCI-X timer is disabled. 1 PCI-X timer is enabled (default).
8–31	TCNT	Number of system clocks to purge a split completion buffer. This occurs when a split completion does not follow a split response.

15.3.2 PCI/X Configuration Header

The *PCI Local Bus Specification* defines the configuration registers contained within the PCI/X configuration header from 0x00 through 0x3F. The *PCI-X Addendum to the PCI Local Bus Specification* defines additional registers beyond 0x3F. Figure 15-25 lists the common PCI/X configuration header as implemented by the MPC8560.

Reserved

				Address Offset (Hex)
Device ID		Vendor ID		00
PCI Bus Status		PCI Bus Command		04
Bus Base Class Code	Subclass Code	Bus Programming Interface	Revision ID	08
BIST Control	Header Type	Bus Latency Timer	Bus Cache Line Size	0C
PCI Configuration and Status Register Base Address Register (PCSRBAR)				10
32-Bit Memory Base Address Register				14
64-Bit Low Memory Base Address Register				18
64-Bit High Memory Base Address Register				1C
64-Bit Low Memory Base Address Register				20
64-Bit High Memory Base Address Register				24
				28
Subsystem ID		Subsystem Vendor ID		2C
				30
			PCI Bus Capability Pointer	34
				38
PCI Bus MAX_LAT	PCI Bus MIN_GNT	PCI Bus Interrupt Pin	PCI Bus Interrupt Line	3C
				40
PCI Bus Arbiter Configuration		PCI Bus Function		44

Figure 15-25. MPC8560 PCI/X Configuration Header

Figure 15-26 lists the additional PCI-X configuration registers.

Reserved

			Address Offset (Hex)
PCI-X Command	PCI-X Next Capability	PCI-X Capability ID	60
PCI-X Status			64

Figure 15-26. PCI-X Additional Configuration Registers

Note that software must be restricted from accessing any reserved or undefined register space on the MPC8560; doing so may hang the device.

Table 15-54 in Section 15.4.2.11.1, “PCI Configuration Space Header,” provides a summary of the PCI configuration header registers. Detailed descriptions of these registers are provided in the *PCI Local Bus Specification*.

15.3.2.1 PCI Vendor ID Register—Offset 0x00

The PCI vendor ID register, shown in Figure 15-27, is used to identify the manufacturer of the part.

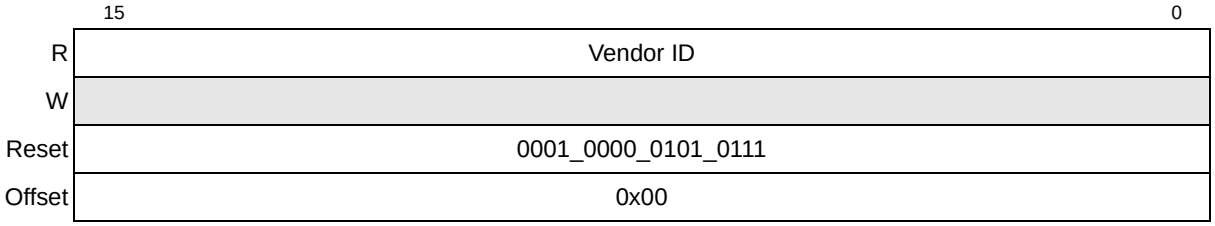


Figure 15-27. PCI Vendor ID Register

Table 15-25 describes PCI vendor ID register fields.

Table 15-25. PCI Vendor ID Register Field Description

Bits	Name	Description
15–0	Vendor ID	0x1057 (Freescale Semiconductor)

15.3.2.2 PCI Device ID Register—Offset 0x02

The PCI device ID register, shown in Figure 15-28, is used to identify the device.

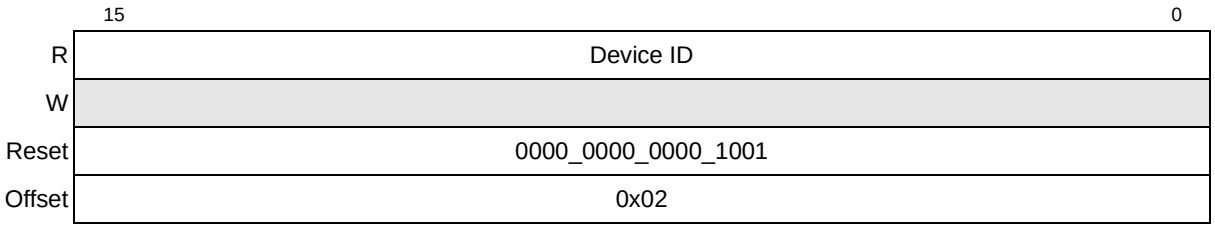


Figure 15-28. PCI Device ID Register

Table 15-26. PCI Device ID Register Field Description

Bits	Name	Description
15–0	Device ID	0x0009

15.3.2.3 PCI Bus Command Register—Offset 0x04

The 2-byte PCI bus command register provides control over the ability to generate and respond to PCI cycles. Table 15-27 describes the bits of the PCI bus command register.

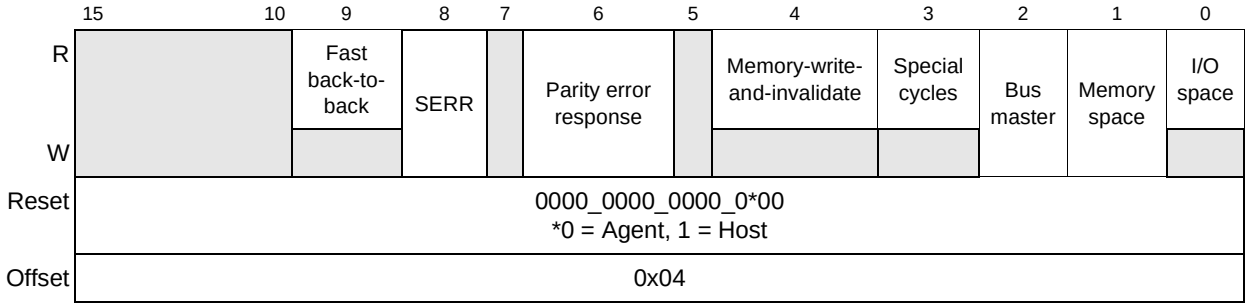


Figure 15-29. PCI Bus Command Register

Table 15-27. PCI Bus Command Register Field Descriptions

Bits	Name	Description
15–10	—	Reserved
9	Fast back-to-back	Hard-wired to 0, indicating that this PCI controller (as a master) does not run fast back-to-back transactions
8	SERR	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	—	Reserved
6	Parity error response	Controls whether this PCI controller responds to parity errors. 0 Parity errors are ignored and normal operation continues. 1 Parity errors cause the appropriate bit in the PCI status register to be set. However, note that errors are reported based on the values set in the PCI error enable and detection registers.
5	—	Reserved
4	Memory-write-and-invalidate	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	Hard-wired to 0, indicating that this PCI controller (as a target) ignores all special-cycle commands
2	Bus master	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) Note that the Bus master bit in the MPC8560's PCI bus command register must be set before any outbound configuration access is attempted.
1	Memory space	Controls whether this PCI controller (as a target) responds to memory accesses. 0 This PCI controller does not respond to PCI memory space accesses. 1 This PCI controller (as a target) responds to PCI memory space accesses.
0	I/O space	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 15-28. Only 2-byte accesses to address offset 0x06 are allowed.

Note that some errors are reported in two bits—one in the PCI bus status register and another in the PCI/X error detect register (ERR_DR). These bits must be cleared separately; that is, clearing one does not clear the other. For example, clearing the ERR_DR[Mstr abort error] does not clear the received master abort bit in the PCI bus status register. In these cases, both bits must be cleared before further error reporting can occur. Refer to Table 15-57 for PCI mode error actions and Table 15-65 for PCI-X mode error actions. Likewise, some errors are enabled by programming two bits—one in the PCI bus command register and another in the PCI/X error enable register (ERR_EN).

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.

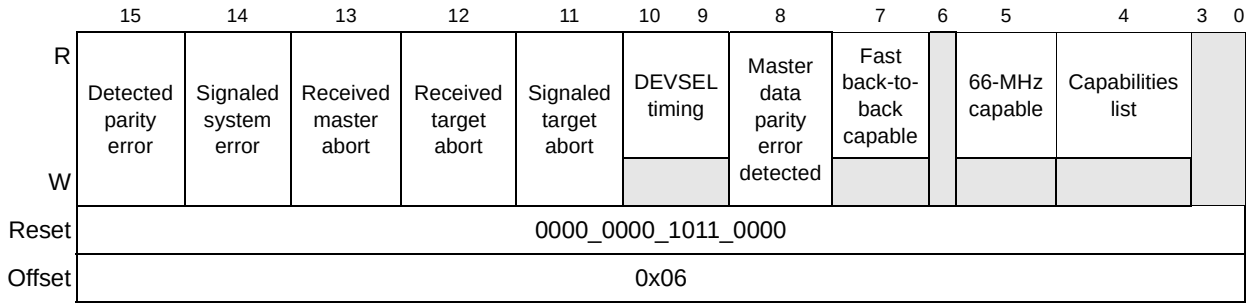


Figure 15-30. PCI Bus Status Register

Table 15-28. PCI Bus Status Register Field Descriptions

Bits	Name	Description
15	Detected parity error	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	Set whenever this PCI controller asserts $\overline{\text{PCI\_SERR}}$.
13	Received master-abort	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	Set whenever a PCI transaction initiated by this PCI controller (excluding a special-cycle) is terminated by a target-abort
11	Signaled target-abort	Set whenever this PCI controller, acting as the PCI target, issues a target-abort to a PCI master
10–9	DEVSEL timing	Hard-wired to 0b00, indicating that this PCI controller uses fast device select timing.

Table 15-28. PCI Bus Status Register Field Descriptions (continued)

Bits	Name	Description
8	Master data parity error detected	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 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	Hard-wired to 1, indicating that this PCI controller (as a target) is capable of accepting fast back-to-back transactions
6	—	Reserved
5	66-MHz capable	Read-only bit indicates that this PCI controller is capable of 66 MHz PCI bus operation.
4	Capabilities List	Hard-wired to 1
3–0	—	Reserved

15.3.2.5 PCI Revision ID Register—Offset 0x08

The PCI Revision ID register is used to identify the revision of the part.

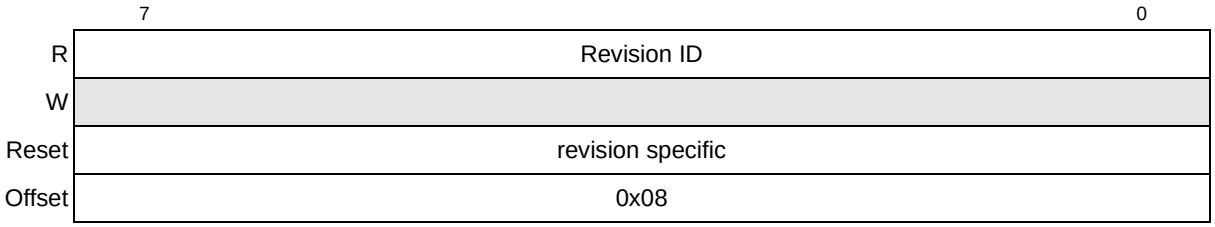


Figure 15-31. PCI Revision ID Register

Table 15-29. PCI Revision ID Register Field Descriptions

Bits	Name	Description
7–0	Revision ID	Revision specific

15.3.2.6 PCI Bus Programming Interface Register—Offset 0x09

Table 15-30 describes the PCI Bus programming interface register (PIR).

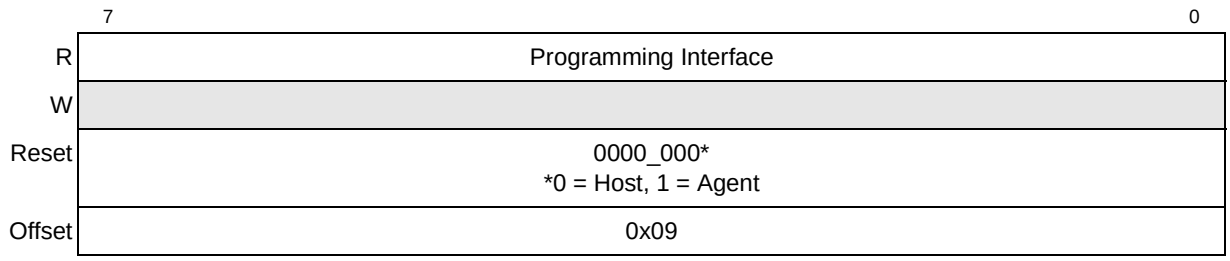


Figure 15-32. PCI Bus Programming Interface Register

Table 15-30. PCI Bus Programming Interface Register Field Description

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

15.3.2.7 PCI Subclass Code Register—Offset 0x0A

Table 15-32 describes the PCI subclass code register (PSCR).

	7	0
R	Subclass Code	
W		
Reset	0010_0000	
Offset	0x0A	

Figure 15-33. PCI Subclass Code Register

Table 15-31. PCI Subclass Code Register Field Description

Bits	Name	Description
7–0	Subclass Code	PowerPC—0x20

15.3.2.8 PCI Bus Base Class Code Register—Offset 0x0B

Table 15-32 describes the PCI bus base class code register (PBCCR).

	7	0
R	Base Class Code	
W		
Reset	0000_1011	
Offset	0x0B	

Figure 15-34. PCI Bus Base Class Code Register

Table 15-32. PCI Bus Base Class Code Register Field Description

Bits	Name	Description
7–0	Base Class Code	Processor—0x0B

15.3.2.9 PCI Bus Cache Line Size Register—Offset 0x0C

Table 15-33 describes the PCI bus cache line size register (PCLSR).

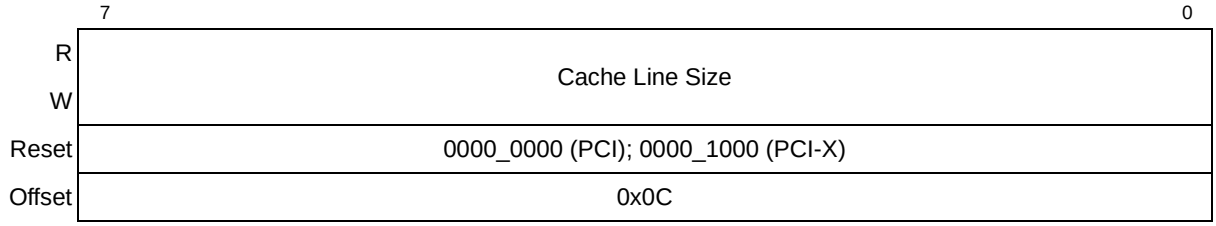


Figure 15-35. PCI Bus Cache Line Size Register (PCLSR)

Table 15-33. PCI Bus Cache Line Size Register (PCLSR) Field Descriptions

Bits	Name	Description
7–0	Cache Line Size	Represents the cache line size of the processor in terms of 32-bit words (eight 32-bit words = 32 bytes). PCLSR 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.

15.3.2.10 PCI Bus Latency Timer Register—0x0D

Table 15-34 describes the PCI latency timer register (PLTR).

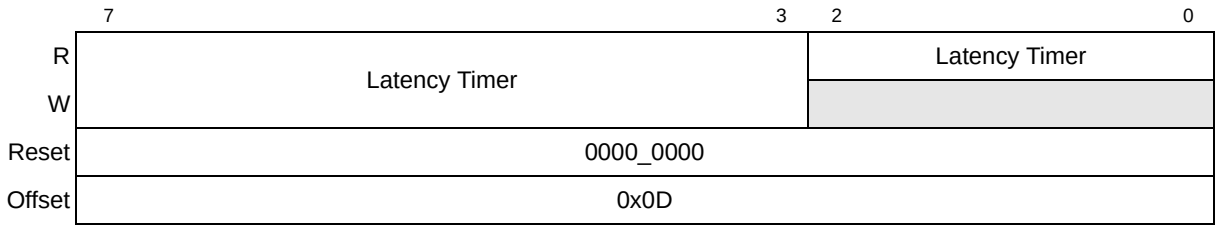


Figure 15-36. PCI Bus Latency Timer Register

Table 15-34. PCI Bus Latency Timer Register Field Descriptions

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.

15.3.2.11 PCI Base Address Registers

A PCI base address register points to the beginnings of each address range to which the device responds by asserting $\overline{\text{PCI\_DEVSEL}}$. The base address register (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 15.3.1.3, “PCI/X ATMU Inbound Registers.” The 32-bit base address register at offset 0x14 corresponds to 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. If one of these registers is read, the corresponding size of ATMU is returned on the PCI bus providing valid window size in the Inbound ATMU window attributes register.

Note that PCSRBAR cannot be updated through the inbound ATMU registers.

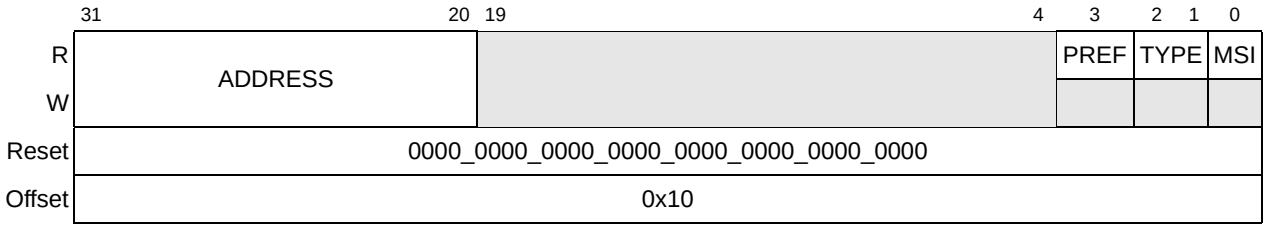


Figure 15-37. PCI Configuration and Status Register Base Address Register (PCSRBAR)

Table 15-35. PCSRBAR Field Descriptions

Bits	Name	Description
31–20	ADDRESS	Indicates the base address at which the inbound configuration/run-time window resides. This window is fixed at 1 Mbyte.
19–4	—	Reserved
3	PREF	Prefetchable
2–1	TYPE	Type. 00—Locate anywhere in 32-bit address space.
0	MSI	Memory space indicator

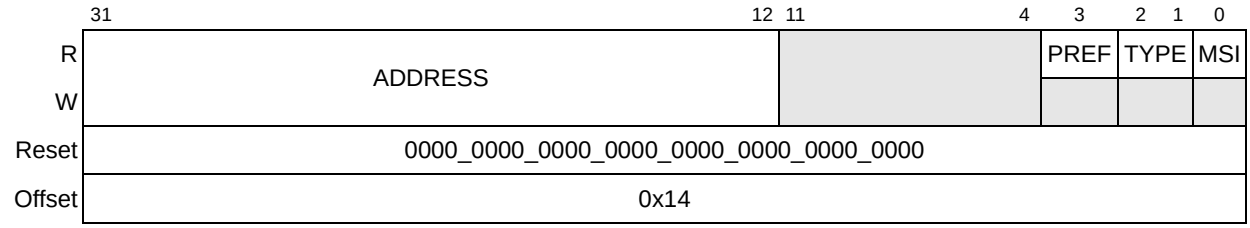


Figure 15-38. 32–Bit Memory Base Address Register

Table 15-36. 32-Bit Memory Base Address Register Field Descriptions

Bits	Name	Description
31–12	ADDRESS	Indicates the base address at which the inbound memory window resides. The number of upper bits that the device allows to be writable is selected through the inbound translation windows.
11–4	—	Reserved. The device allows a 4 Kbyte window minimum.
3	PREF	Prefetchable
2–1	TYPE	Type: 00—Locate anywhere in 32-bit address space.
0	MSI	Memory Space Indicator.

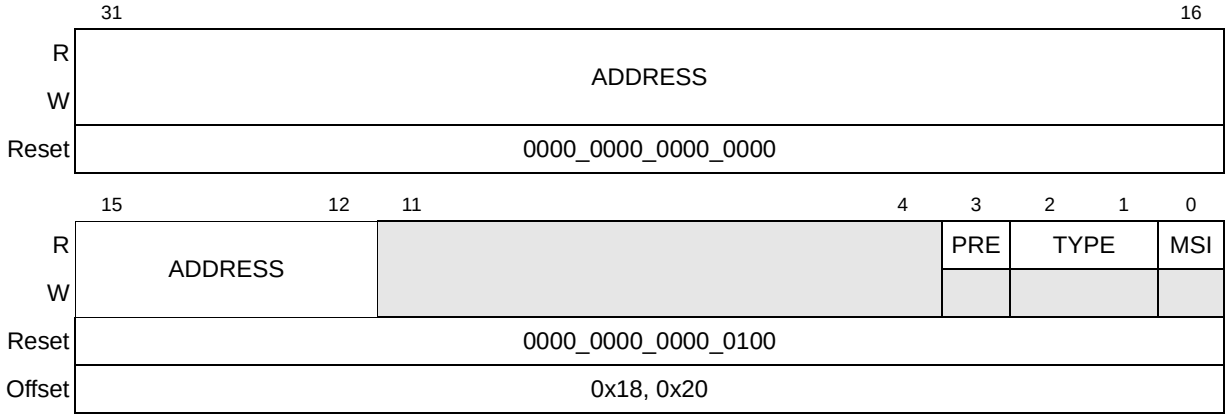


Figure 15-39. 64-Bit Low Memory Base Address Register

Table 15-37. 64-Bit Low Memory Base Address Register Field Descriptions

Bits	Name	Description
31–12	ADDRESS	Indicates the base address at which the inbound memory window resides. The number of upper bits that the device allows to be writable is selected through the inbound translation windows.
11–4	—	Reserved. The device allows a 4 Kbyte window minimum.
3	PREF	Prefetchable
2–1	TYPE	Type: 10—Locate anywhere in 64-bit address space.
0	MSI	Memory space indicator

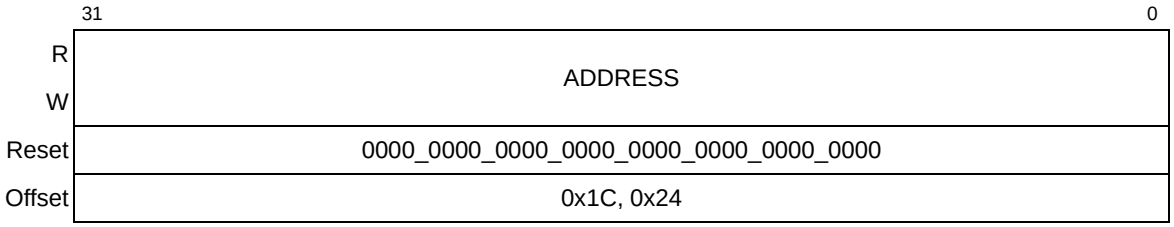


Figure 15-40. 64-Bit High Memory Base Address Register

Table 15-38. 64-Bit High Memory Base Address Register Field Description

Bits	Name	Description
31–0	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.

15.3.2.12 PCI Subsystem Vendor ID Register

The PCI subsystem vendor ID register is used to identify the subsystem.

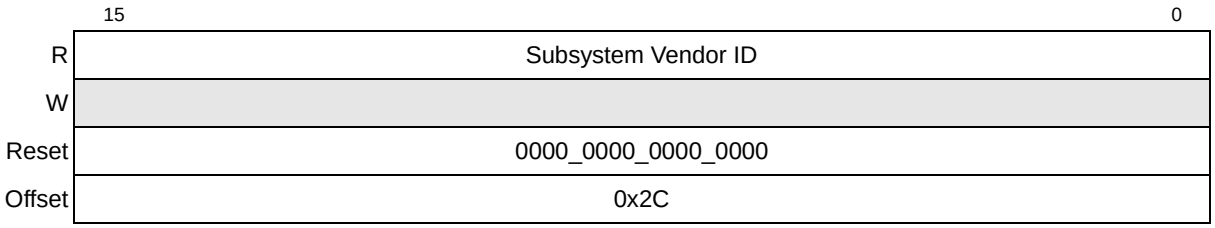


Figure 15-41. PCI Subsystem Vendor ID Register

Table 15-39. PCI Subsystem Vendor ID Register Field Description

Bits	Name	Description
15–0	Subsystem Vendor ID	0x0000

15.3.2.13 PCI Subsystem ID Register

The PCI Subsystem ID register is used to identify the subsystem.

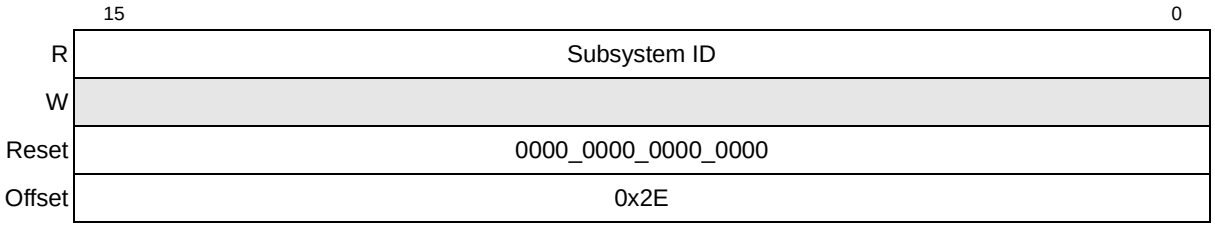


Figure 15-42. PCI Subsystem ID Register

Table 15-40. PCI Subsystem ID Register Field Description

Bits	Name	Description
15–0	Subsystem ID	0x0000

15.3.2.14 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 15-41.

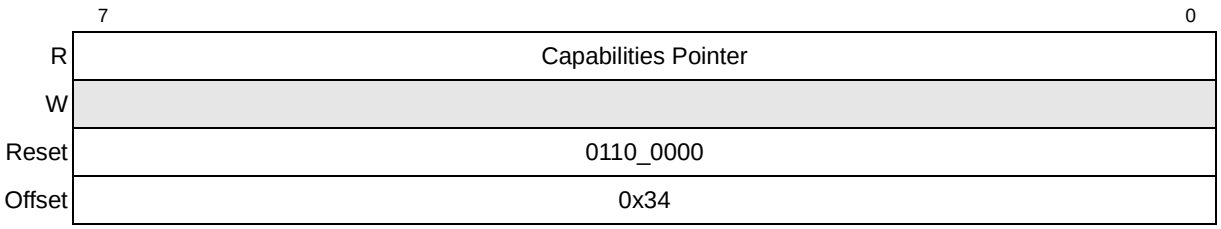


Figure 15-43. PCI Bus Capabilities Pointer Register

Table 15-41. PCI Bus Capabilities Pointer Register Field Description

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.

15.3.2.15 PCI Bus Interrupt Line Register

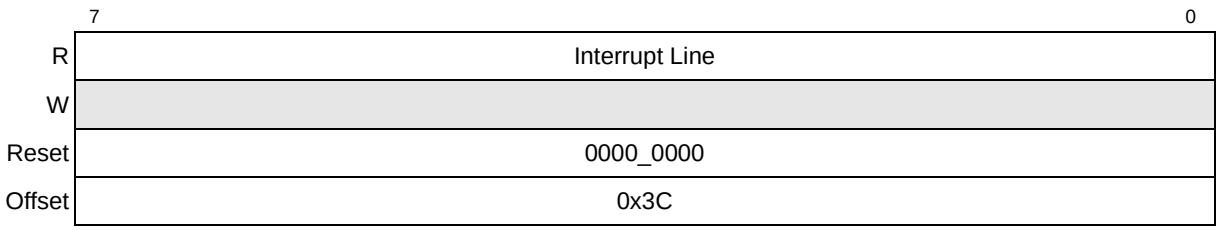


Figure 15-44. PCI Bus Interrupt Line Register

Table 15-42. PCI Bus Interrupt Line Register Field Description

Bits	Name	Description
7–0	Interrupt Line	This register is used to communicate interrupt line routing information (hard-wired to zero).

15.3.2.16 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/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/X) can be routed.

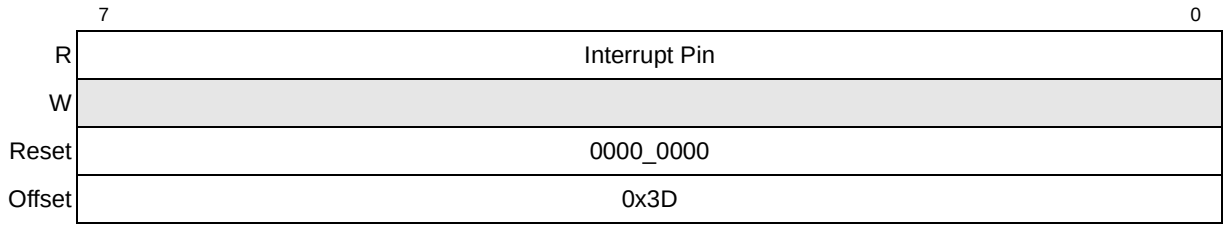


Figure 15-45. PCI Bus Interrupt Pin Register

Table 15-43. PCI Bus Interrupt Pin Register Field Description

Bits	Name	Description
7–0	Interrupt Pin	No PCI_INT pin selected

15.3.2.17 PCI Bus Minimum Grant (MIN_GNT) Register

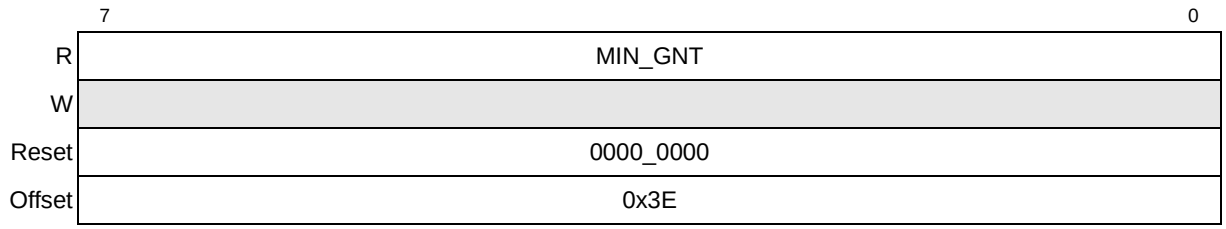


Figure 15-46. PCI Bus Minimum Grant Register

Table 15-44. PCI Bus Minimum Grant Register Field Description

Bits	Name	Description
7–0	MIN_GNT	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)

15.3.2.18 PCI Bus Maximum Latency (MAX_LAT) Register

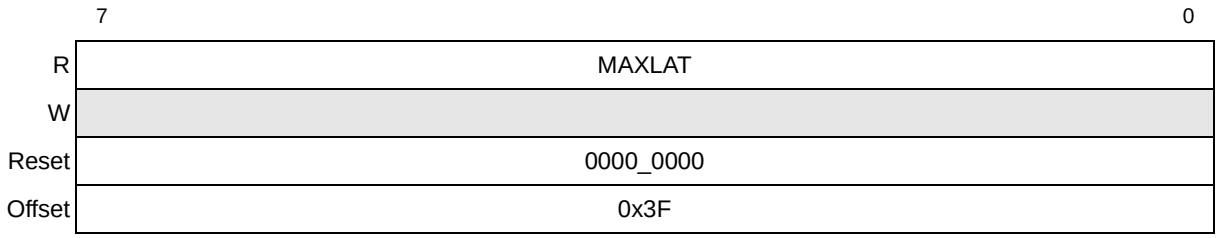


Figure 15-47. PCI Bus Maximum Latency Register

Table 15-45. PCI Bus Maximum Latency Register Field Description

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)

15.3.2.19 PCI Bus Function Register (PBFR)

The 2-byte PCI bus function register is used to determine how different features of the PCI interface are configured. This register is at PCI configuration space at offset 0x44.

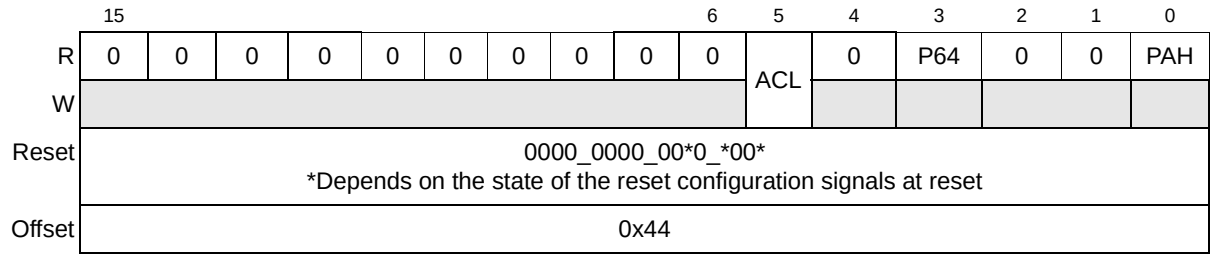


Figure 15-48. PCI Bus Function Register

Table 15-46. PCI Bus Function Register Field Descriptions

Bits	Name	Description
15–6	—	Reserved
5	ACL	Agent configuration 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 LA27 (cfg_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

Table 15-46. PCI Bus Function Register Field Descriptions (continued)

Bits	Name	Description
3	P64	PCI 64-bit configuration. Read-only. Indicates the reset value of the PCI 64-bit configuration signal, PCI_REQ64. 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 agent/host. Read-only. Indicates the reset value of the PCI host/agent configuration signal, LWE3. 0 PCI interface is in host mode 1 PCI interface is in agent mode

15.3.2.20 PCI Bus Arbiter Configuration Register (PBACR)

The PCI bus arbiter configuration register is used to determine the configuration of the PCI bus arbiter.

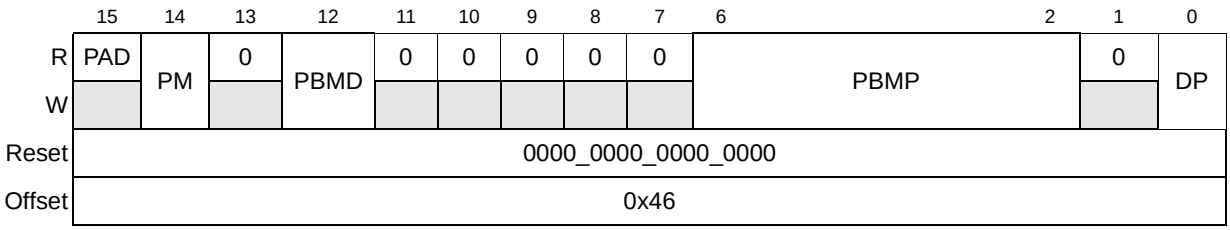


Figure 15-49. PCI Bus Arbiter Configuration Register

Table 15-47. PCI Bus Arbiter Configuration Register Field Descriptions

Bits	Name	Description
15	PAD	PCI arbiter disable. Determines if the device is the PCI arbiter on the PCI bus or not. The reset state is determined by the inverse of the PCI_GNT2 configuration input signal when reset is released. 0 Device is the PCI arbiter. 1 Device is not the PCI arbiter. Device presents its request on PCI_REQ0 to the external arbiter and receives its grant on 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 this device.
13	—	Reserved
12	PBMD	PCI broken master disable. 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

Table 15-47. PCI Bus Arbiter Configuration Register Field Descriptions (continued)

Bits	Name	Description
6–2	PBMP	PCI bus master priorities. Determines arbitration priority given to different masters on the PCI bus. Bit 6 corresponds to the priority of the master sourcing PCI_REQ0; bit 2 corresponds to the priority of the master sourcing PCI_REQ4. 0 Master *n* is low priority. 1 Master *n* is high priority.
1	—	Reserved
0	DP	Device priority. Determines this device's arbitration priority. 0 Device is low priority. 1 Device is high priority.

15.3.2.21 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*.

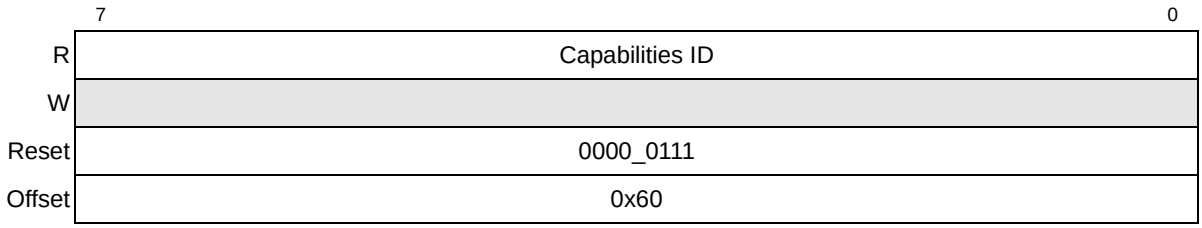


Figure 15-50. PCI-X Next Capabilities ID Register

Table 15-48. PCI-X Next Capabilities ID Register Field Descriptions

Bits	Name	Description
7–0	Capabilities ID	This register identifies this item in the capabilities list as a PCI-X device having PCI-X registers.

15.3.2.22 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 15-49.

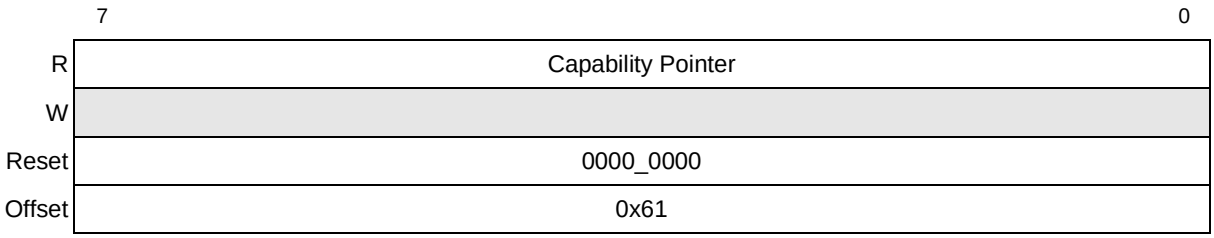


Figure 15-51. PCI-X Capability Pointer Register

Table 15-49. PCI-X Capability Pointer Register Field Description

Bits	Name	Description
7–0	Capability Pointer	Specifies the byte offset in the configuration space containing the first item in the capabilities list.

15.3.2.23 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 15-50 describes the bits of the PCI-X command register.

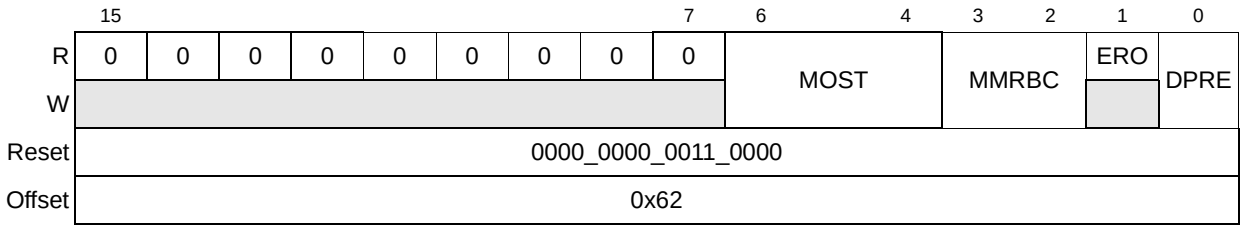


Figure 15-52. PCI-X Command Register

Table 15-50. PCI-X Command Register Field Descriptions

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 (read-only)
0	DPRE	Data parity error recovery enable (supported)

15.3.2.24 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. In agent mode, the device and bus number are updated by the inbound configuration write transaction except for the inbound configuration write to PCI-X status register. The definition of each bit is given in Table 15-51. Only 4-byte accesses to address offset 0x64 are allowed.

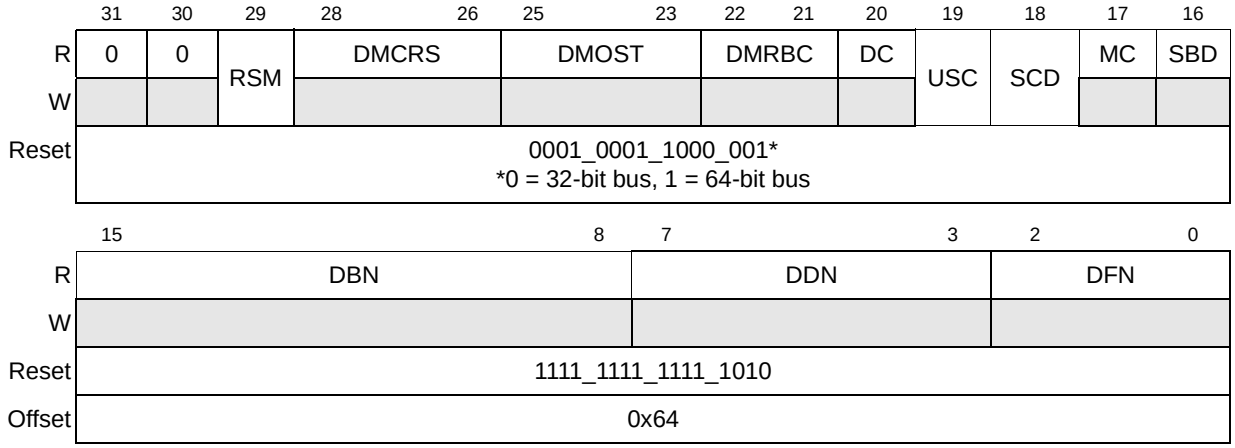


Figure 15-53. PCI-X Status Register

Table 15-51. PCI-X Status Register Field Descriptions

Bits	Name	Description
31–30	—	Reserved
29	RSM	Receive split completion error message (write-1-to-clear)
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/X interface.

15.4.1 PCI/X Bus Arbitration

PCI/X bus arbitration is access-based. Bus masters must arbitrate for each access performed on the bus. The PCI/X bus uses a central arbitration scheme where each master has a 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/X 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/X bus masters. The on-chip PCI/X arbiter functions in both host and agent modes, or it can be disabled to allow for an external PCI/X arbiter.

A configuration signal ($\overline{\text{PCI\_GNT2}}$) sampled at the negation of $\overline{\text{HRESET}}$ determines if the on-chip PCI/X arbiter is enabled (high) or disabled (low). The state of the reset signal is reflected in bit 15 (read-only) of the PCI/X bus arbitration control register (PBACR[PAD]). Note that PAD is the inverse of the arbiter configuration signal; that is, when PAD = 0 the arbiter is enabled, and when PAD = 1 the arbiter is disabled. See Chapter 4, “Reset, Clocking, and Initialization,” for more information on the reset configuration signals.

If the on-chip PCI/X 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, the internal request/grant pair for the internal master state machine of the MPC8560 governs processor, DMA, and RapidIO accesses to the PCI/X interface. If the on-chip PCI/X 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/X arbiter that arbitrates between external PCI/X masters and the internal PCI/X bus master of the MPC8560.

15.4.1.1 PCI/X Bus Arbiter Operation

The on-chip PCI/X 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/X 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 1 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) × 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 15-54, 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 15-54, 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.

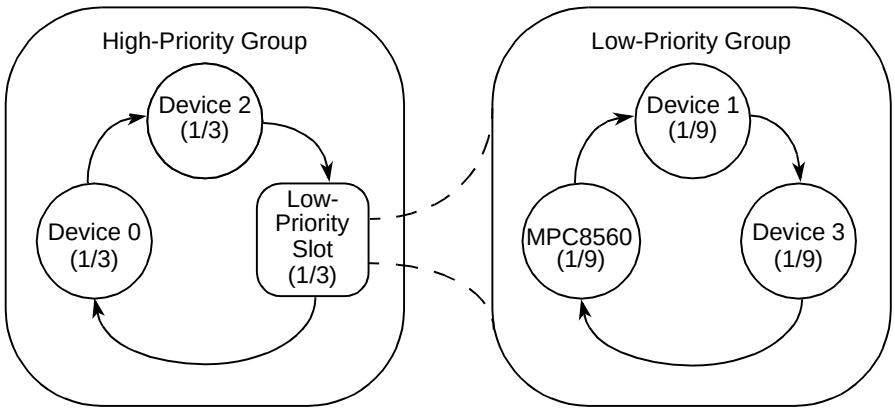


Figure 15-54. PCI/X Arbitration Example

15.4.1.2 PCI/X Bus Parking

When no device is using or requesting the bus, the PCI/X 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/X 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/X bus. If PBACR[PM] = 0 (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] = 1, the bus is parked on the MPC8560.

15.4.1.3 Broken Master Lock-Out (PCI only)

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 (0 = enabled, 1 = disabled). Note that the PCI-X bus arbiter does not support the broken master lockout function.

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, arbitration logic cannot 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 the doze or nap 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 15-55, Figure 15-56, Figure 15-57, and Figure 15-58 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 Integrated Processor Hardware Specifications* for specific setup and hold times.

15.4.2.1 Basic Transfer Control

The basic PCI bus transfer mechanism is a burst, composed of an address phase followed by one or more data phases. Fundamentally, all PCI data transfers are controlled by three signals—PCI_FRAME (frame), PCI_IRDY (initiator ready), and PCI_TRDY (target ready). An initiator asserts PCI_FRAME to indicate the beginning of a PCI bus transaction and negates PCI_FRAME to indicate the end of a PCI bus transaction. An initiator negates PCI_IRDY to force wait cycles. A target negates PCI_TRDY to force wait cycles.

The PCI bus is considered idle when both PCI_FRAME and PCI_IRDY are negated. The first clock cycle in which 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 PCI_C/BE[3:0] during the address phase of a PCI transaction. The bus command indicates to the target the type of transaction the initiator is requesting. Table 15-52 describes the PCI bus commands implemented by the MPC8560.

Table 15-52. 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	A read (implicitly addressing the system interrupt controller). Only one device on the PCI bus should respond to this command; others ignore it. See Section 15.4.2.12.1, "Interrupt-Acknowledge Transactions," for more information.
0001	Special cycle	Yes	No	Provides a way 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	Accesses agents mapped into the PCI I/O space.
0011	I/O-write	Yes	No	Accesses agents mapped into the PCI I/O space.
0100	Reserved[1]	No	No	—
0101	Reserved[1]	No	No	—
0110	Memory-read	Yes	Yes	Accesses either local memory or agents mapped into PCI memory space, depending on the address. When a PCI master issues this 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	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	Access 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 details.
1011	Configuration-write	Yes	Agent mode only	
1100	Memory-read-multiple	Yes	Yes	Similar to the memory-read command, but also causes a prefetch of the next cache line (32 bytes).
1101	Dual-address-cycle	Yes	Yes	Used to transfer a 64-bit address (in two 32-bit address cycles) to 64-bit addressable devices.
1110	Memory-read-line	Yes	Yes	Indicates that an initiator is requesting the transfer of an entire cache line. This occurs only when the processor is performing a burst read. Note that these processors perform burst reads only when the appropriate cache is enabled and the transaction is not cache-inhibited.
1111	Memory-write-and-invalidate	No	Yes	Indicates that an initiator is transferring an entire cache line; 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/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 looks for accesses in the address range that the device has been assigned. For subtractive decoding, one device on the bus looks 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 15-53. The other two PCI_AD[1:0] possibilities (0b01 and 0b11) are reserved. As an initiator, the MPC8560 always encodes PCI_AD[1:0] = 00 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] = 01 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 15-53. 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 bus 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 bus 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. PCI_PERR has a turnaround cycle on the fourth clock cycle 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; ‘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 $\overline{\text{PCI\_FRAME}}$. During the address phase, PCI_AD[63:0] contains a valid address and PCI_C/$\overline{\text{BE}}$[7:0] contains a valid bus command.

The first data phase of a read transaction requires a turnaround cycle. This allows the transition from the initiator driving 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 $\overline{\text{TRDY}}$ 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 $\overline{\text{PCI\_DEVSEL}}$ is asserted.

During the data phase, the PCI_C/$\overline{\text{BE}}$[7:0] signals indicate which byte lanes are involved in the current data phase. A data phase may consist of a data transfer and wait cycles. The PCI_C/$\overline{\text{BE}}$[7:0] signals remain actively driven for both reads and writes from the first clock of the data phase through the end of the transaction.

A data phase completes when data is transferred, which occurs when both $\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ are asserted on the same clock edge. When either $\overline{\text{PCI\_IRDY}}$ or $\overline{\text{PCI\_TRDY}}$ is negated, a wait cycle is inserted and no data is transferred. The initiator indicates the last data phase by negating $\overline{\text{PCI\_FRAME}}$ when $\overline{\text{PCI\_IRDY}}$ is asserted. The transaction is considered complete when data is transferred in the last data phase.

Figure 15-55 illustrates a PCI single-beat read transaction.

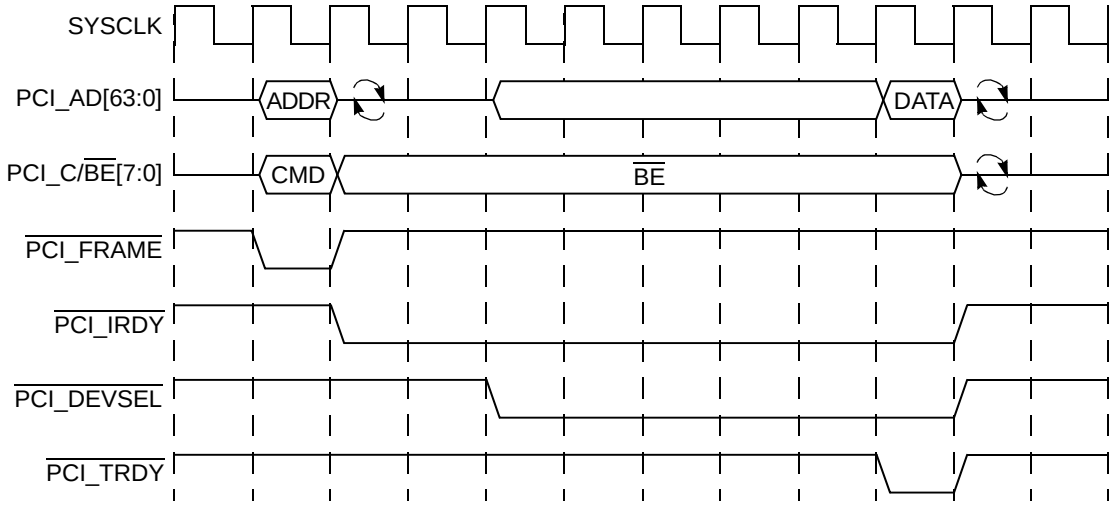


Figure 15-55. PCI Single-Beat Read Transaction

Figure 15-56 illustrates a PCI burst read transaction.

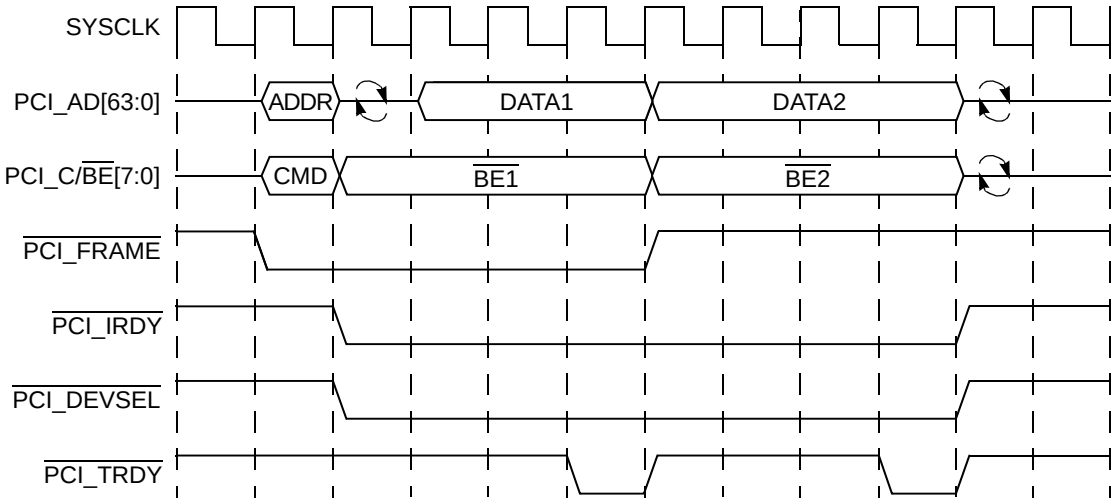


Figure 15-56. PCI Burst Read Transaction

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/$\overline{\text{BE}}$[7:0] signals, even if the initiator is not ready to provide valid data ($\overline{\text{PCI\_IRDY}}$ negated).

Figure 15-57 illustrates a PCI single-beat write transaction.

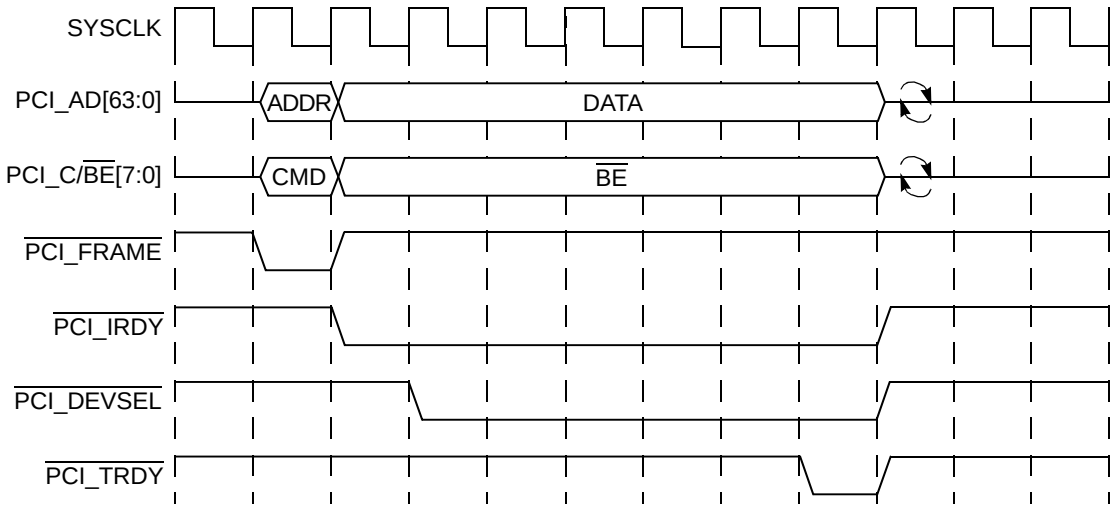


Figure 15-57. PCI Single-Beat Write Transaction

Figure 15-58 illustrates a PCI burst write transaction.

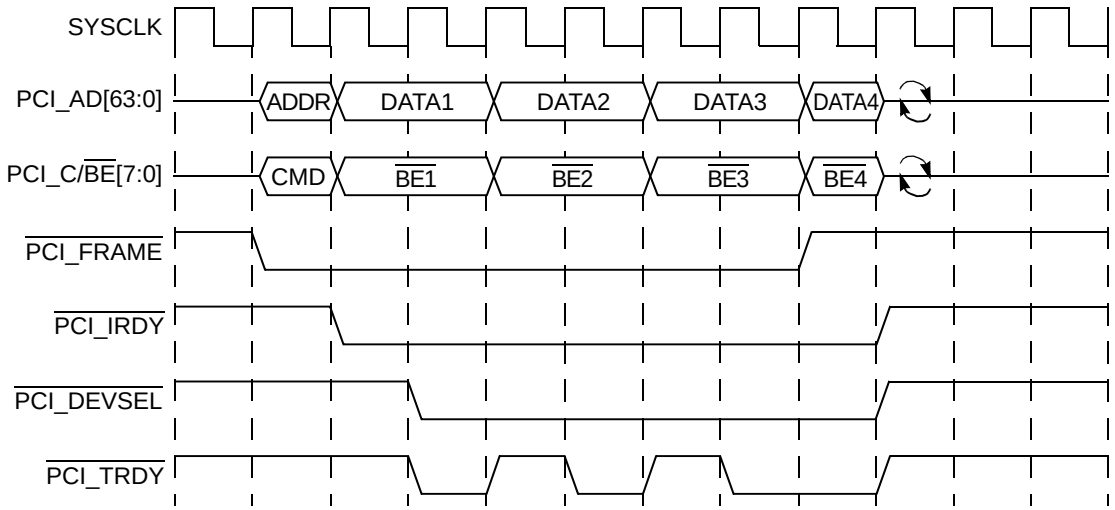


Figure 15-58. PCI Burst Write Transaction

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 bus 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.

15.4.2.8.2 Target-Initiated Termination

By asserting PCI_STOP, 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—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—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—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 cannot 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.”

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 bus status register and the signaled target-abort bit (bit 11) of the target’s bus 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 15-59 shows several target-initiated terminations.

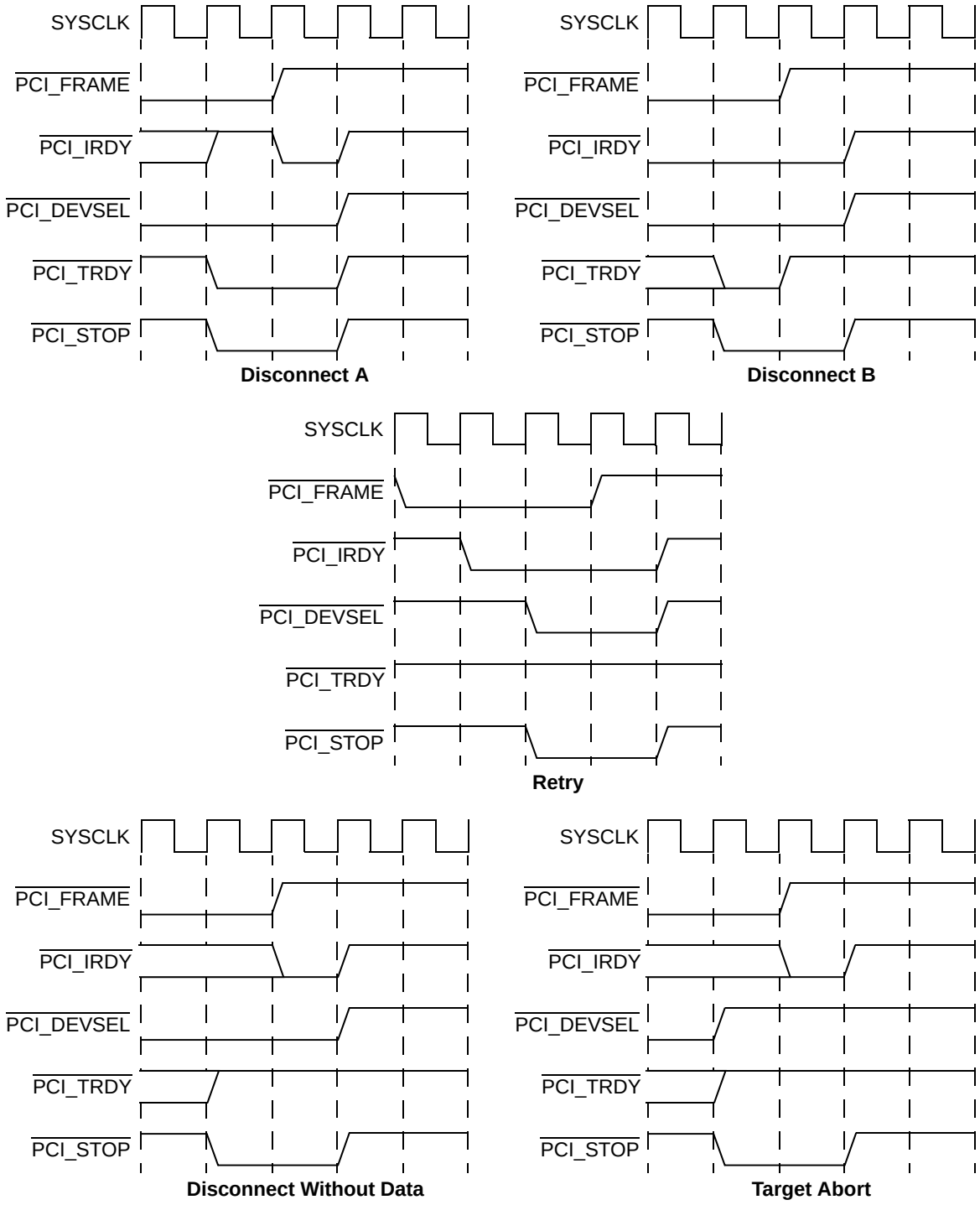


Figure 15-59. PCI Target-Initiated Terminations

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.

For the case of the local processor, DAC generation depends on the setting of the POTEARx. If the POTEARx are programmed with nonzero values and a transaction from the local processor core hits in one of the outbound windows, a DAC transaction is generated on the PCI bus with the translated lower 32-bit addresses. Refer to Section 15.3.1.2, “PCI/X ATMU Outbound Registers,” for more information.

Figure 15-60 shows the timing sequence of the PCI signals for single-beat DAC reads.

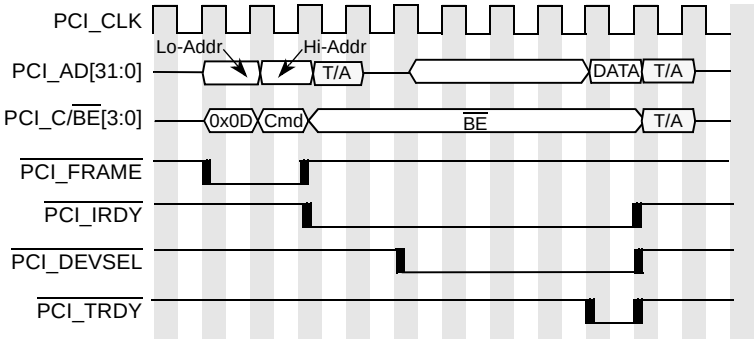


Figure 15-60. DAC Single-Beat Read Example

The timing for a DAC burst read is shown in Figure 15-61.

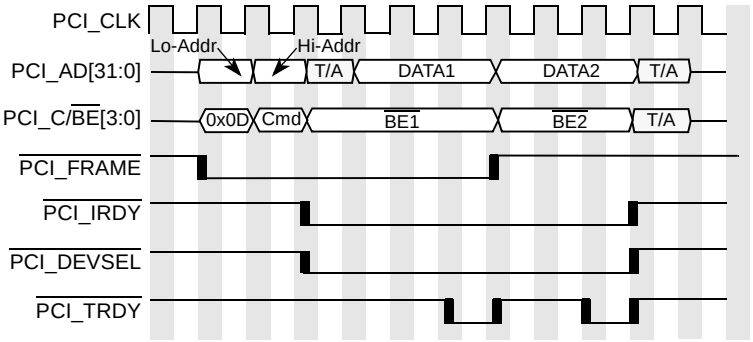


Figure 15-61. DAC Burst Read Example

Figure 15-62 and Figure 15-63 show single-beat DAC writes and burst DAC writes.

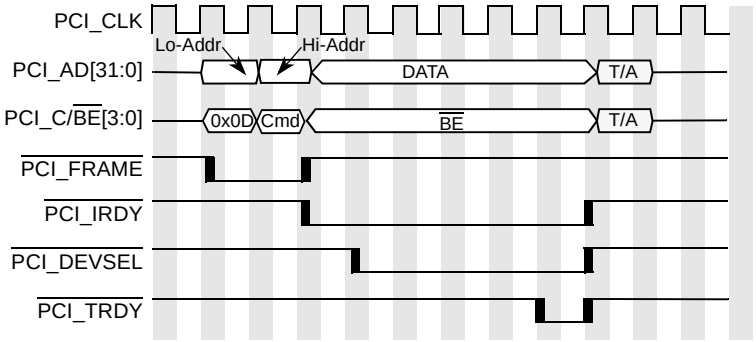


Figure 15-62. DAC Single-Beat Write Example

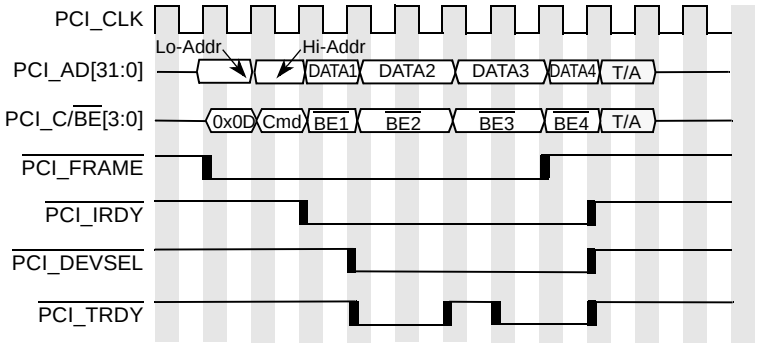


Figure 15-63. DAC Burst Write Example

Figure 15-64 shows the timing sequence of the PCI signals for 64-bit DAC reads.

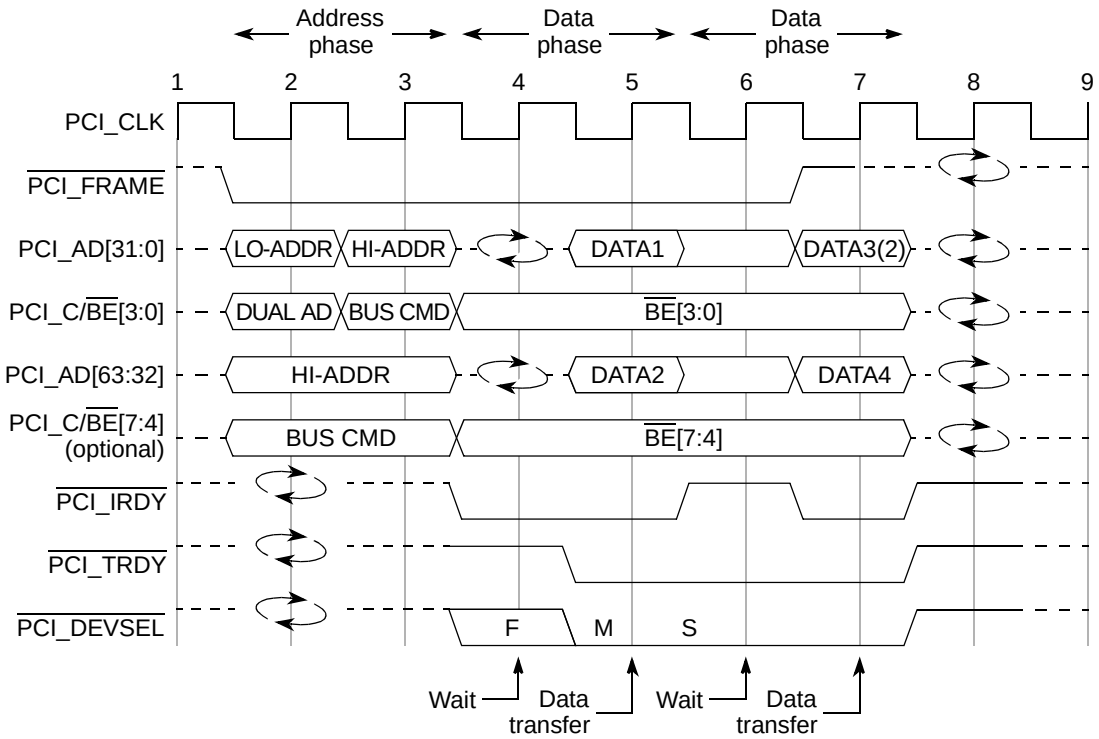


Figure 15-64. 64-Bit Dual Address Read Cycle

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

Figure 15-65 shows the predefined header for all PCI devices.

				Address Offset
Device ID		Vendor ID		0x00
Status		Command		0x04
Class Code			Revision ID	0x08
BIST	Header Type	Latency Timer	Cache Line Size	0x0C
Base Address Registers				0x10 0x14 0x18 0x1C 0x20 0x24
Reserved				0x28
Subsystem ID		Subsystem Vendor ID		0x2C
Expansion ROM Base Address				0x30
Reserved				0x34
Reserved				0x38
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	0x3C

Figure 15-65. Standard PCI Configuration Header

The first 64 bytes of the 256-byte configuration space consists of a predefined header that every PCI device must support. The first 16 bytes of the predefined header are defined the same for all PCI 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 15-65. 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/X Configuration Header.”

Table 15-54 summarizes the configuration header registers. Detailed descriptions of these registers are provided in the *PCI Local Bus Specification*, Rev. 2.2.

Table 15-54. PCI Configuration Space Header Summary

Offset	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
0x0E	Header type	Bits 0–6 identify the layout of bytes 0x10–0x3F; bit 7 indicates a multifunction device. The most common header type (0x00) is shown in Figure 15-65 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 (read-only for MPC8560)
0x2E	Subsystem ID	Identifies the subsystem ID (read-only for MPC8560)
0x30	Expansion ROM base address	Base address and size information for expansion ROM contained in an add-on board
0x34, 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 in Host Mode

To access the configuration space, a 32-bit value must be written to the PCI CFG_ADDR register that specifies the target PCI bus, the target device on that bus, and the configuration register to be accessed within that device. Note that the Bus Master bit in the MPC8560's PCI bus command register must be set before an outbound configuration access is attempted. Device 0 on PCI/X bus 0 is the MPC8560 itself; thus, device 0, bus 0 is used to access the internal PCI/X configuration header.

When the MPC8560 detects an access to PCI CFG_DATA, it checks the enable flag and the device number in the PCI 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. 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. 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.

See Section 15.3.1.1.1, "PCI/X Configuration Address Register (CFG_ADDR)," for details on PCI CFG_ADDR and Section 15.3.1.1.2, "PCI/X Configuration Data Register (CFG_DATA)," for details on PCI CFG_DATA.

Note that because all PCI/X registers are intrinsically little-endian, in the following examples, the data in the configuration register is shown in little-endian order. PowerPC processor accesses to the PCI 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 byte lane redirection from the little-endian PCI bus is performed internally.

Example: Configuration sequence, 4-byte data read from the revision ID/standard programming interface/subclass code/class code registers at address offset 0x08 of the PCI/X configuration header (device 0 on the PCI/X bus 0 is the MPC8560 itself).

```
Initial values:
        r0 contains 0x8000_0008
        r1 contains CCSRBAR + 0x0_8000 (Address of PCI CFG_ADDR register)
        r2 contains CCSRBAR + 0x0_8004 (Address of PCI CFG_DATA register)
        r3 contains 0xFFFF_FFFF
        Register at 0x08 contains 0x0B20_0002 (0x0B to 0x08)

Code sequence:
        stw r0, 0 (r1)
        lwbrx r3, 0 (r2)

Results:
        Address CCSRBAR + 0x0_8000 contains 0x8000_0008
        Register r3 contains 0x0B20_0002
```

Example: Configuration sequence, 4-byte data write to PCI/X register at address offset 0x14 of Device 1 on PCI/X bus 0.

```
Initial values:
        r0 contains 0x8000_0814
        r1 contains CCSRBAR + 0x0_8000 (Address of PCI CFG_ADDR register)
        r2 contains CCSRBAR + 0x0_8004 (Address of PCI CFG_DATA register)
        r3 contains 0x1122_3344
        Register at 0x14 contains 0xFFFF_FFFF (0x17 to 0x14)
```

```
Code sequence:
        stw r0, 0 (r1)  // Update PCI CFG_ADDR register to point to
                        //register offset 0x14 of device 1.
        stwbrx r3, 0 (r2)

Results:
        Address CCSRBAR + 0x0_8000 contains 0x8000_0814
        Register at 0x14 contains 0x1122_3344 (0x17 to 0x14)
```

Example: Configuration sequence, 2-byte data write to PCI register at address offset 0x1C of Device 1 on PCI/X bus 0.

```
Initial values:
        r0 contains 0x8000_081C
        r1 contains CCSRBAR + 0x0_8000
        r2 contains CCSRBAR + 0x0_8004
        r3 contains 0xDDCC_BBAA
        Register at 0x1C contains 0xFFFF_FFFF (0x1F to 0x1C)

Code sequence:
        stw r0, 0 (r1)
        sthbrx r3, 0 (r2)

Results:
        Address CCSRBAR + 0x0_8000 contains 0x8000_081C
        Register at 0x1C contains 0xFFFF_BBAA (0x1F to 0x1C)
```

15.4.2.11.3 PCI Configuration in Agent and Agent Lock Modes

In general, agents should not access the configuration space of other external PCI devices. Configuration of agents is a function usually reserved for the host. When the MPC8560 is in agent mode, it responds to remote host-generated PCI/X configuration cycles. This occurs when a configuration command is decoded along with the IDSEL input signal being asserted. When the MPC8560 is in agent lock mode, it retries all externally-generated PCI/X configuration cycles until the ACL bit in the PCI bus function register (0x44) is set. See Section 15.5.1, “Power-On Reset Configuration Modes,” for more information.

In either agent or agent lock mode, access to the internal PCI/X configuration header by the processor core is handled as described in Section 15.4.2.11.2, “Accessing the PCI Configuration Space in Host Mode,” using device = 0 and bus = 0 in PCI CFG_ADDR to indicate the internal PCI/X header.

15.4.2.11.4 PCI Type 0 Configuration Translation

Figure 15-66 shows the PCI type 0 translation function performed on the contents of the PCI CFG_ADDR register to the PCI_AD[31:0] signals on the PCI bus during the address phase of the configuration cycle.

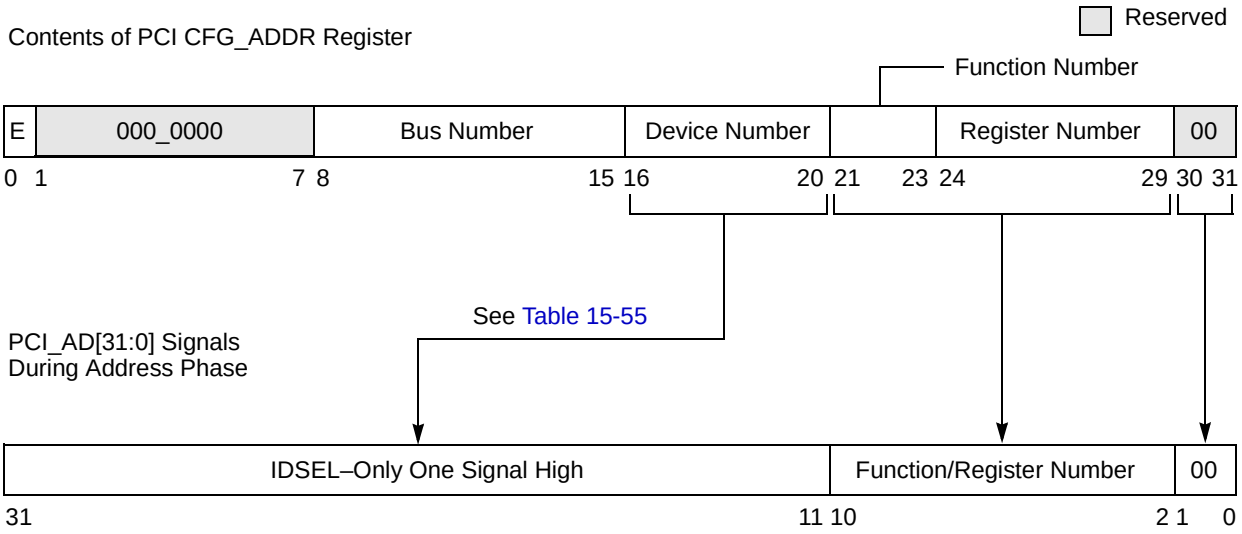


Figure 15-66. PCI Type 0 Configuration Translation

For PCI type 0 configuration cycles, the MPC8560 translates the device number field of the PCI CFG_ADDR register into a unique IDSEL signal for up to 21 different devices. Each device connects its IDSEL input to one of the PCI_AD[31:11] signals. For PCI type 0 configuration cycles, the MPC8560 translates the device number to AD*n* as shown in Table 15-55.

Table 15-55. PCI Type 0 Configuration—Device Number to AD*n* Translation

Device Number		AD*n* Used for IIDSEL	Device Number		AD*n* Used for IIDSEL
Binary	Decimal		Binary	Decimal	
0b0_0000–0b0_1001	0–9	—	0b1_0101	21	AD21
0b0_1010	10	AD31	0b1_0110	22	AD22
0b0_1011	11	AD11	0b1_0111	23	AD23
0b0_1100	12	AD12	0b1_1000	24	AD24
0b0_1101	13	AD13	0b1_1001	25	AD25
0b0_1110	14	AD14	0b1_1010	26	AD26
0b0_1111	15	AD15	0b1_1011	27	AD27
0b1_0000	16	AD16	0b1_1100	28	AD28
0b1_0001	17	AD17	0b1_1101	29	AD29
0b1_0010	18	AD18	0b1_1110	30	AD30
0b1_0011	19	AD19	0b1_1111 [1]	31	—
0b1_0100	20	AD20			

[1] A device number of all ones indicates a PCI special-cycle or interrupt-acknowledge transaction.

For PCI 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 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 PCI_AD[31:0] and PCI_C/$\overline{\text{BE}}$[3:0] one clock cycle before the assertion of $\overline{\text{PCI\_FRAME}}$.

15.4.2.11.5 Type 1 Configuration Translation

For type 1 translations, the MPC8560 copies the 30 high-order bits of the PCI 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 PCI CFG_ADDR and PCI CFG_DATA registers described in Section 15.4.2.11.3, "PCI Configuration in Agent and Agent Lock Modes."

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 PCI CFG_DATA, it checks the enable flag and the device number in PCI CFG_ADDR. 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 PCI_TRDY is asserted. The size of the interrupt vector returned is indicated by the value driven on PCI_C/BE[7:0].

The MPC8560 also can generate PCI interrupt-acknowledge transactions directly. Reads from PCI 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 PCI CFG_DATA, it checks the enable flag and the device number in PCI CFG_ADDR. 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 other configuration cycles for which the bus number does not match.

Aside from the special-cycle command (PCI_C/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 15-56.

Table 15-56. 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. It is unnecessary to assert PCI_DEVSEL in response to a special-cycle command. The initiator of the special-cycle transaction can insert wait states, but because there is no specific target, the special-cycle message and optional data field are valid on the first clock PCI_IRDY is asserted. All special-cycle transactions are terminated by master-abort; however, the master-abort bit in the initiator's bus 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. 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/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/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/BE[7:4] and PCI_C/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 15-67.

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.

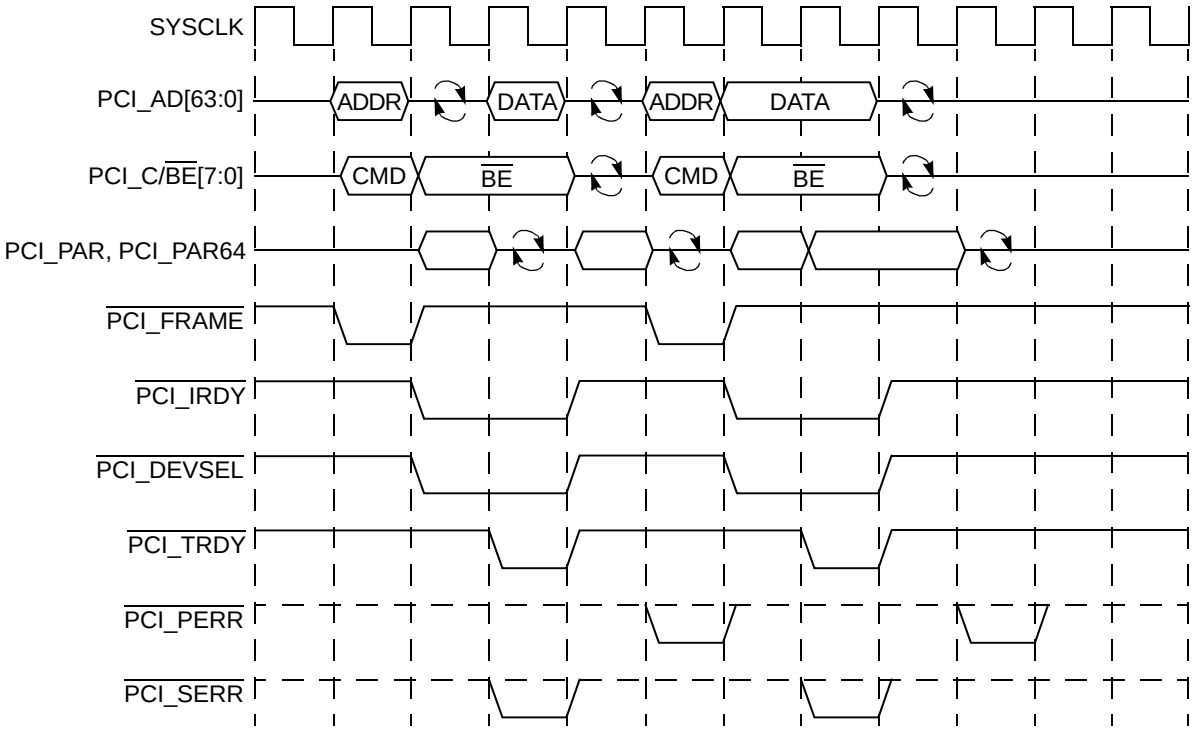


Figure 15-67. PCI Parity Operation

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—PCI_PERR and PCI_SERR. The PCI_PERR signal is used exclusively to report data parity errors on all transactions except special cycles. The PCI_SERR signal is used for other error signaling including address parity errors and data parity errors on special-cycle transactions; it may also be used to signal other system errors.

Note that some errors are enabled by programming two bits—one in the PCI bus command register and another in the PCI/X error enable register (ERR_EN). Likewise, some errors are reported in two bits—one in the PCI bus status register and another in the PCI/X error detect register (ERR_DR). These bits must be cleared separately; that is, clearing one does not clear the other. For example, clearing the ERR_DR[Mstr abort error] does not clear the received master abort bit in the PCI bus status register. In these cases, both bits must be cleared before further error reporting can occur.

Table 15-57 shows the actions taken for each kind of error.

Table 15-57. PCI Mode Error Actions

PCI Error Type	Error Detect Register Bit	PCI Bus Status Register Bit	Comment
PCI Outbound Read			
Received $\overline{\text{SERR}}$ at any phase	Rcvd $\overline{\text{SERR}}$ error	—	No data transferred
Received parity error for data phase	Mstr $\overline{\text{PERR}}$ error	Detected parity error, Master data parity error detected	No data transferred
Master abort	Mstr abort error	Received master abort	No data transferred
Target abort	Trgt abort error	Received target abort	No data transferred
Memory space violation	ORMSV error	—	No data transferred. Only 8 bytes are requested in PCI bus
PCI Outbound Write			
Received $\overline{\text{SERR}}$ related to address phase	Rcvd $\overline{\text{SERR}}$ error	—	May float AD bus to avoid contention
Received $\overline{\text{SERR}}$ related to data phase	Rcvd $\overline{\text{SERR}}$ error	—	
Received $\overline{\text{PERR}}$ (data phase)	Mstr $\overline{\text{PERR}}$ error	Master data parity error detected	
Master abort	Mstr abort error	Received master abort	
Target abort	Trgt abort error	Received target abort	
Memory space violation	OWMSV error	—	Only 8 bytes transferred.
PCI Inbound Read			
Detected parity error for address phase	Addr Parity error	Detected parity error, Signaled system error	Float AD bus
Detected parity error on upper address bus for address phase (SAC or DAC)	—	—	No PAR64 check during address phase
Received $\overline{\text{SERR}}$ at any phase	Rcvd $\overline{\text{SERR}}$ error	—	
Received $\overline{\text{PERR}}$ (Data phase)	Trgt $\overline{\text{PERR}}$ error	—	
Internal error	Trgt abort error	Signaled target abort	
PCI Inbound Write			
Detected parity error for address phase	Addr Parity error	Detected parity error, Signaled system error	Cache line purged
Detected parity error on upper address bus for address phase (SAC or DAC)	—	—	No PAR64 check during address phase
Received $\overline{\text{SERR}}$ at any phase	Rcvd $\overline{\text{SERR}}$ error	—	
Detected parity error for data phase	Trgt $\overline{\text{PERR}}$ error	Detected parity error	Cache line purged

15.4.3 PCI-X Bus Protocol

The PCI-X bus protocol is an enhancement of the PCI bus protocol that features a backward-compatible, higher-bandwidth, 32- or 64-bit multiplexed address/data bus at frequencies up to 133 MHz. The most significant of these enhancements are the following:

- Slightly different command types define burst and DWORD transaction types
- A new attribute phase for each transaction provides more information about the transaction including a byte count, sequence ID, and handling instructions (for ordering and cacheability)
- Restricted wait state and disconnection rules
- A new split transaction, consisting of a split response termination and one or more split completions, replaces the delayed transaction functionality from the PCI bus protocol.

The following sections describe the new functionality of the PCI-X bus protocol.

15.4.3.1 PCI-X Terminology

The *PCI-X Addendum to the PCI Local Bus Specification* introduces terminology that is helpful when describing aspects of the protocol. This terminology is introduced in context in the following sections. However, three terms that are important to keep in mind throughout these sections are defined as follows:

Sequence	A sequence is one or more transactions associated with carrying out a single logical transfer by a requester. Each transaction in the same sequence carries the same unique sequence ID.
Requester	A requester is an initiator that first introduces a transaction into the PCI-X domain.If a transaction is terminated with a split response, the requester becomes the target of the subsequent split completion.
Completer	A completer is the device addressed by a transaction (other than a split completion transaction). If a target terminates a transaction with a split response, the completer becomes the initiator of the subsequent split completion.

15.4.3.2 PCI-X Command Encodings

The PCI-X protocol uses a slightly different set of command encodings for PCI_C/$\overline{\text{BE}}$[3:0] during the address phase of a PCI-X transaction. The bus command indicates to the target the type of transaction the initiator is requesting. Table 15-52 describes the PCI-X bus commands implemented by the MPC8560.

Table 15-58. PCI-X Command Encodings

PCI_C/BE[3:0]	PCI -X Bus Command	Length/ Byte-Enable Usage	MPC8560 Supports as Initiator	MPC8560 Supports as Target	Definition
0000	Interrupt-acknowledge	4 bytes/ Attribute	Yes	No	This command is a read (implicitly addressing the system interrupt controller). Only one device on the PCI bus should respond to this command; other devices ignore it. See Section 15.4.2.12.1, "Interrupt-Acknowledge Transactions."
0001	Special cycle	4 bytes/ Attribute	Yes	No	Provides a way 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	4 bytes/ Attribute	Yes	No	Accesses agents mapped into the PCI I/O space.
0011	I/O-write	4 bytes/ Attribute	Yes	No	Accesses agents mapped into the PCI I/O space.
0100	Reserved[1]	—	—	—	—
0101	Reserved[1]	—	—	—	—
0110	Memory-read-DWORD	4 bytes/ Attribute	Yes	Yes	Accesses 4-bytes in either local memory or agents mapped into PCI memory space, depending on the address.
0111	Memory-write	Burst/ Data phase	Yes	Yes	Accesses either local memory or agents mapped into PCI memory space, depending on the address.
1000	Reserved. (Aliased to memory-read-block)	Burst/ None	No	Aliased	Reserved for use in future revisions of the PCI-X specification. Current targets must treat this command as if it were a a memory-read-block.
1001	Reserved. (Aliased to memory-write-block)	Burst/ None	No	Aliased	Reserved for use in future revisions of the PCI-X specification. Current targets must treat this command as if it were a a memory-write-block.
1010	Configuration-read	4 bytes/ Attribute	Yes	Agent mode only	Accesses the configuration space of a PCI-X agent. See Section 15.4.3.7, "PCI-X Configuration Transactions," for more detail on PCI-X configuration cycles.
1011	Configuration-write	4 bytes/ Attribute	Yes	Agent mode only	Accesses the configuration space of a PCI-X agent. See Section 15.4.3.7, "PCI-X Configuration Transactions," for more detail on PCI-X configuration cycles.
1100	Split-completion	Burst/ None	Yes (as completer)	Yes (as requester)	Similar to the memory-write command, but the transaction is associated with a previous read or write transaction that was terminated with a split response.

Table 15-58. PCI-X Command Encodings (continued)

PCI_C/BE[3:0]	PCI -X Bus Command	Length/ Byte-Enable Usage	MPC8560 Supports as Initiator	MPC8560 Supports as Target	Definition
1101	Dual-address-cycle	—	Yes	Yes	Indicates the transfer a 64-bit address (in two 32-bit address cycles) to 64-bit addressable devices. The actual transaction command is transfered either on PCI_C/$\overline{BE}$[3:0] in the second address phase for a 32-bit bus or on PCI_C/$\overline{BE}$[7:4] for a 64-bit bus.
1110	Memory-read-block	Burst/ None	Yes	Yes	Indicates that an initiator is requesting the transfer of a block of data.
1111	Memory-write-block	Burst/ None	Yes	Yes	Indicates that an initiator is transferring a block of data.

[1] Reserved command encodings are reserved for future use. The MPC8560 does not respond to these commands.

15.4.3.3 PCI-X Attribute Phase

PCI-X defines a new attribute phase that occurs one clock cycle after the address phase. During the attribute phase, a specially encoded attribute is driven by the initiator on PCI_C/$\overline{BE}$[3:0] and AD[31:0]. PCI_C/$\overline{BE}$[3:0] provide either the upper four bits of the byte count for burst transactions or byte enables for DWORD transactions. Figure 15-68 shows PCI_AD[31:0] attributes for burst and DWORD transactions.

The attributes for split completion transactions are described in Section 15.4.3.6, “PCI-X Split Transactions,” and the attributes for PCI-X configuration transactions are described in Section 15.4.3.7, “PCI-X Configuration Transactions.”

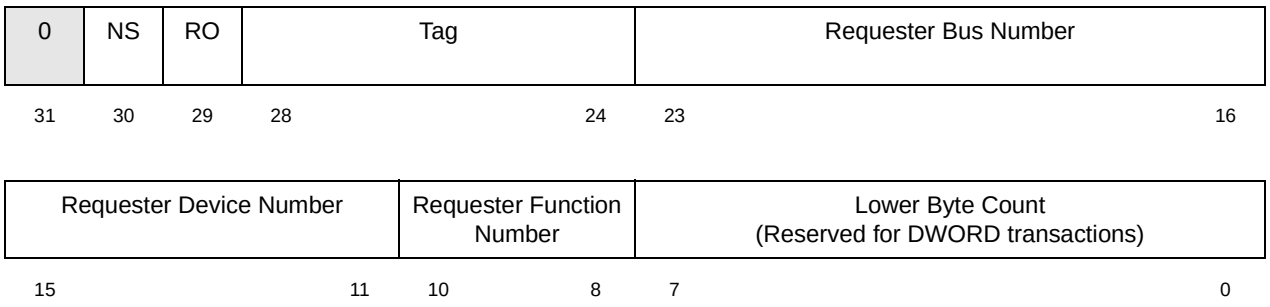


Figure 15-68. AD[31:0] During PCI-X Burst/DWORD Attribute Phase

Table 15-59 describes the fields of the attribute for burst and DWORD transactions.

Table 15-59. Burst/DWORD Transaction Attribute Summary

AD	Name	Description
31	—	Reserved. Must be 0
30	NS	No snoop. This attribute indicates whether the sequence should be snooped for coherency. 0 The requester cannot guarantee that the locations between the starting and ending address, inclusive, of this sequence are not stored in any cache in the system. Thus, all potentially affected caches should be snooped. 1 The requester guarantees that the locations between the starting and ending address, inclusive, of this sequence are not stored in any cache in the system. Thus, snooping is not necessary.
30	RO	Relaxed ordering. This attribute indicates whether the requester requires a transaction to be strictly ordered. In general, devices set the RO attribute for payload sequences and clear the RO attribute for control and status sequences. Refer to the *PCI-X Addendum to the PCI Bus Specification* for a complete discussion of usage models for relaxed transaction ordering. 0 The requester requires that the transaction remain in strict order. 1 The requester guarantees that the transaction is not required to remain in strict order.
28:24	Tag	The tag attribute identifies up to 32 unique sequences from a single initiator. Tag is also a component of the sequence ID.
23:16	Requester Bus Number	The requester bus number attribute is supplied by the bus number field in the PCI-X status register. The requester bus number is also a component of the sequence ID.
15:11	Requester Device Number	The requester device number attribute is supplied by the device number field in the PCI-X status register. The requester bus number is also a component of the sequence ID.
10:8	Requester Function Number	The requester function number attribute is supplied by the function number field in the PCI-X status register. The requester bus number is also a component of the sequence ID.
7:0	Lower Byte Count	For a burst transaction attribute phase, PCI_C/$\overline{\text{BE}}$[3:0] is concatenated with AD[7:0] to create a 12-bit byte count. The byte count indicates the number of bytes the initiator plans to transfer in the remainder of the sequence. For DWORD transactions, the transaction has an implicit size of up to 4-bytes. Therefore, for DWORD transactions, there is no byte count and AD[7:0] is reserved. PCI_C/$\overline{\text{BE}}$[3:0] contain the individual byte enables for DWORD transactions.

15.4.3.4 PCI-X Transactions

Figure 15-69 shows a typical PCI-X write transaction. To highlight some of the difference with the PCI protocol, compare this to the PCI write transaction shown in Figure 15-58.

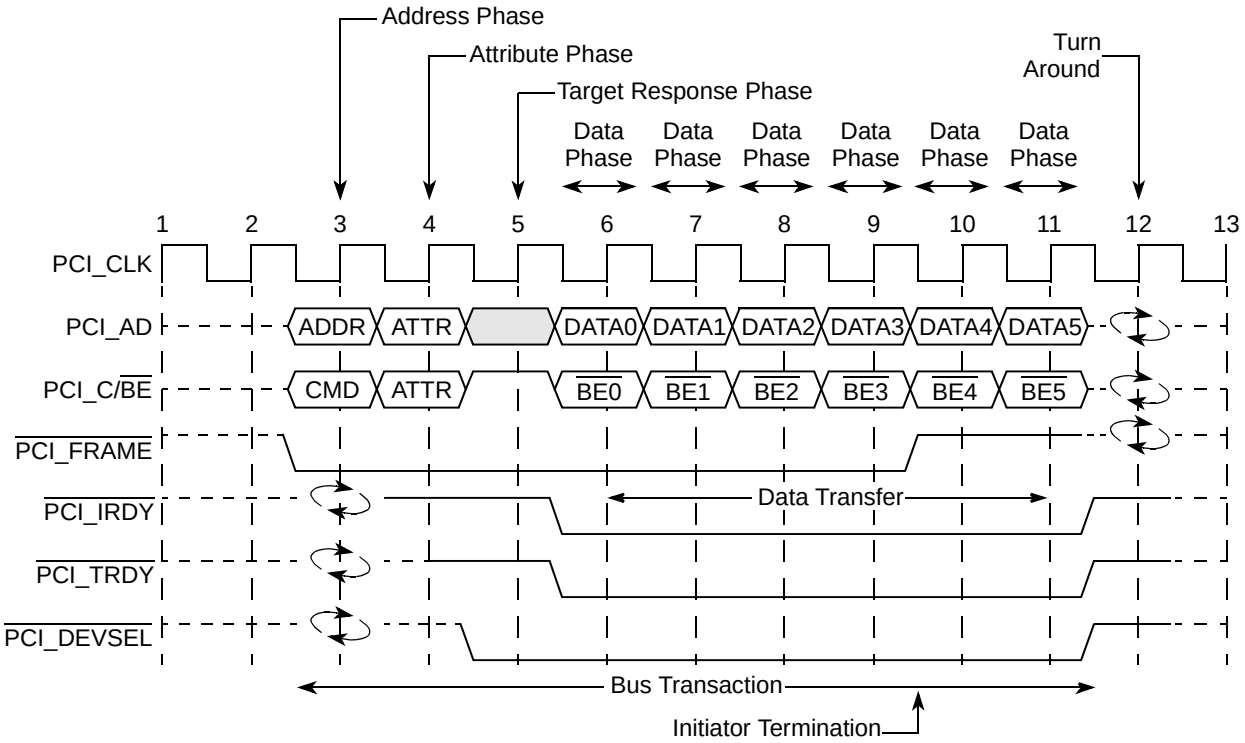


Figure 15-69. Typical PCI-X Write Transaction

Figure 15-70 shows a the following burst write transactions: memory write, memory write block, and split completion transactions. Signals preceded by “s1_” indicate a signal that is internal to the device after the signal has been sampled. For the memory write block and split completion transactions the PCI_C/$\overline{\text{BE}}$[7:0] signals are driven high by the initiator on every data phase. Note that there are no target initial wait states for this example.

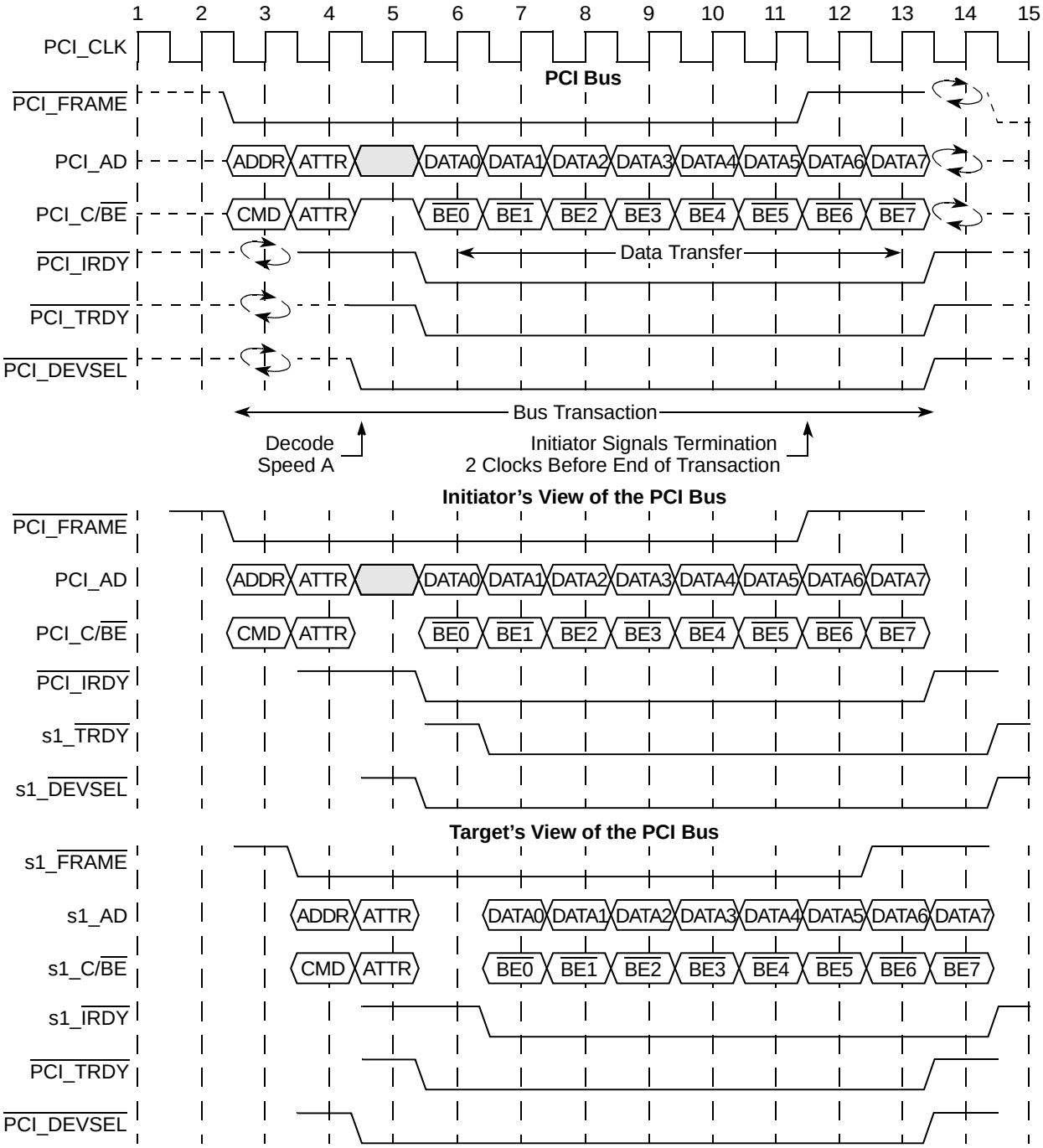


Figure 15-70. Burst Memory Write Transaction

Figure 15-71 shows a memory read block transaction. Signals preceded by "s1_" indicate a signal that is internal to the device after the signal has been sampled. Note that there are no target initial wait states for this example.

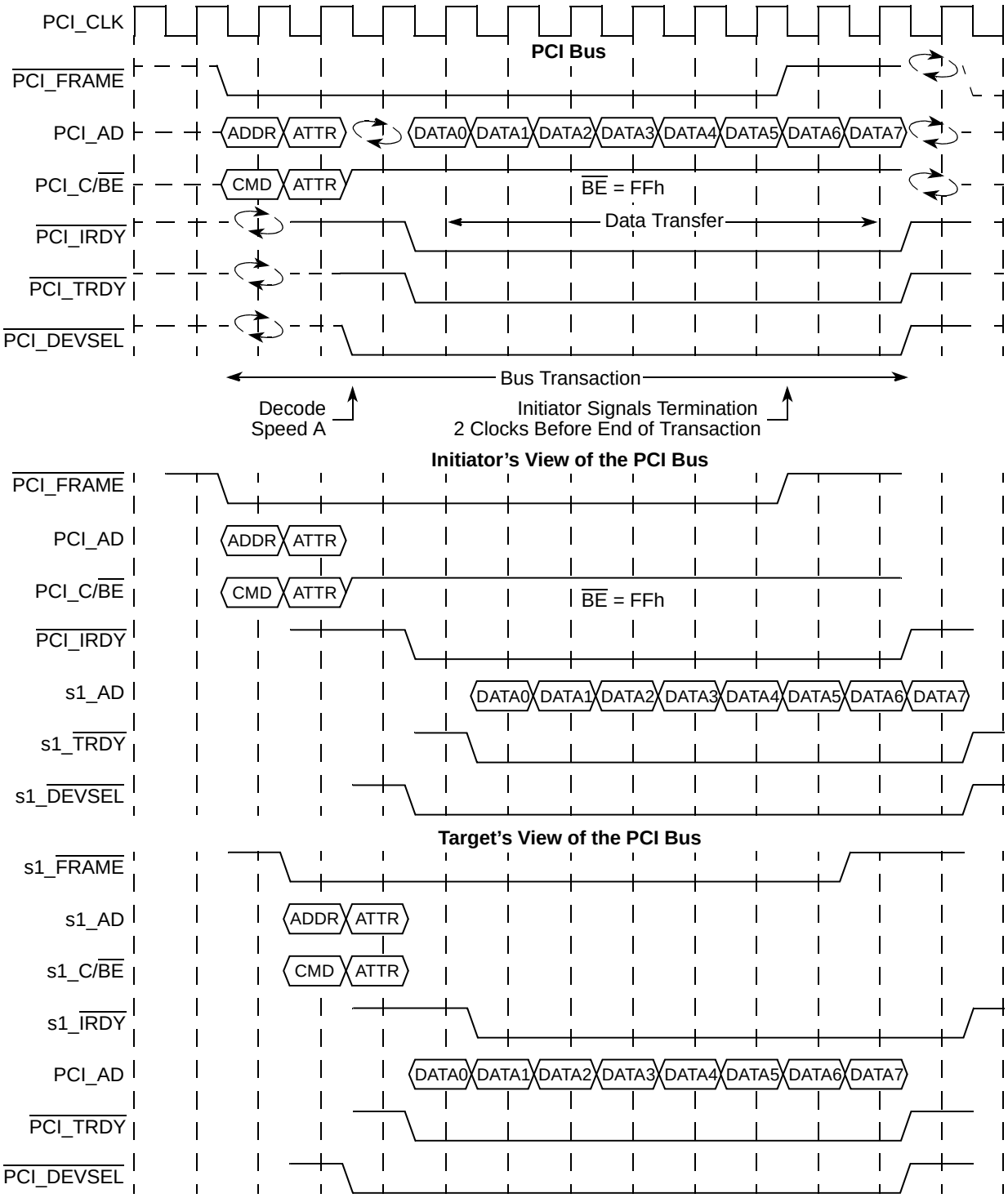


Figure 15-71. Memory Read Block Transaction

Figure 15-72 shows a DWORD write transaction such as an I/O write. Note that there are no target initial wait states for this example.

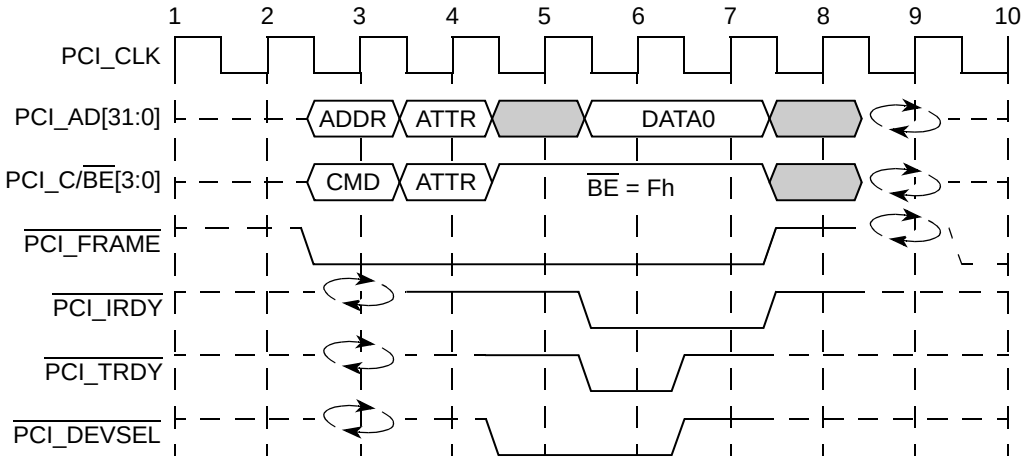


Figure 15-72. DWORD (4-Byte) Write Transaction

Figure 15-73 shows a DWORD read transaction that can be used for a memory read DWORD or an I/O read transaction. This example has two target initial wait states before data is transfered.

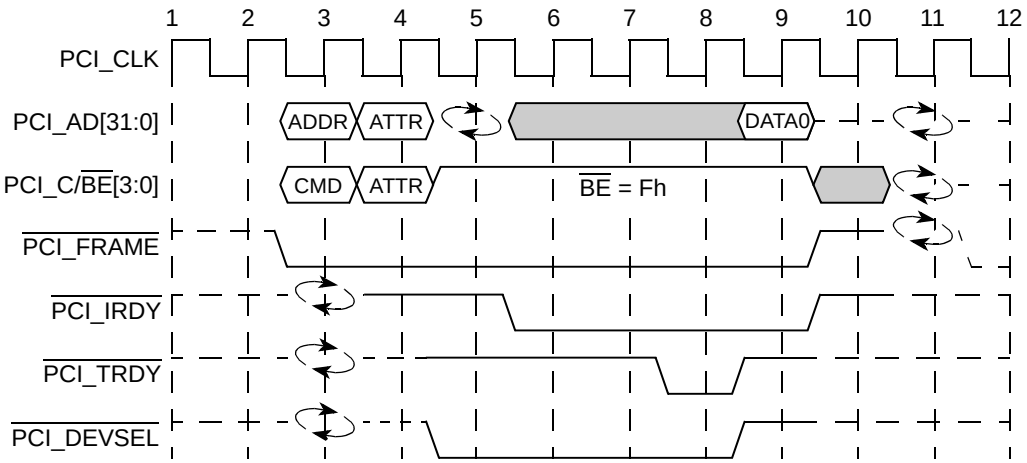


Figure 15-73. DWORD (4-Byte) Read Transaction

15.4.3.5 PCI-X Wait State and Termination Rules

PCI-X adds the following restrictions to wait state and disconnect rules:

- Initiators are not permitted to insert wait states.
- Targets are not permitted to insert wait states after the initial data phase; that is, targets are only allowed to insert wait states on the initial data phase.

- Initiators and targets can end burst transactions as follows:
  - After a burst data transfer starts and the target signals it can accept more than one data phase, the transaction can only be stopped in one of the following ways:
    - Target or initiator disconnect at an allowable disconnect boundary (ADB). An ADB is a naturally-aligned, 128-byte address; that is, an address whose lower 7 bits are zeros.
    - The transaction byte count is satisfied
    - Target abort
    - Burst transactions can start on any byte address. Both the initiator and the target can disconnect on any ADB. A target is also permitted to disconnect a burst transaction after a single data phase.

In PCI-X mode, the MPC8560 always terminates an inbound write transaction with a target-disconnect at the ADB. Also in PCI-X mode, in addition to the target-retry conditions listed in Section 15.4.2.8.2, “Target-Initiated Termination,” the MPC8560 may retry a subsequent inbound transaction if a previous outbound read transaction was terminated by a master-abort, a split-completion error message, or a target-abort.

15.4.3.6 PCI-X Split Transactions

The basic sequence for a split transaction is as follows:

1. The initiator (requester) requests the bus and the arbiter grants the bus to the requester.
2. The requester initiates a transaction (other than a split completion).
3. The target (completer) communicates its intent to split the transaction by using a split response termination.
4. The completer requests the bus and the arbiter grants the bus to the completer
5. The completer initiates a split completion transaction. Split completions are routed back to original requester across bridges by using requester’s bus, device and function numbers in the split completion address.

15.4.3.6.1 Split Response

Figure 15-74 shows a split response to a read transaction. Note that the read could be either a memory read block or a memory read DWORD transaction.

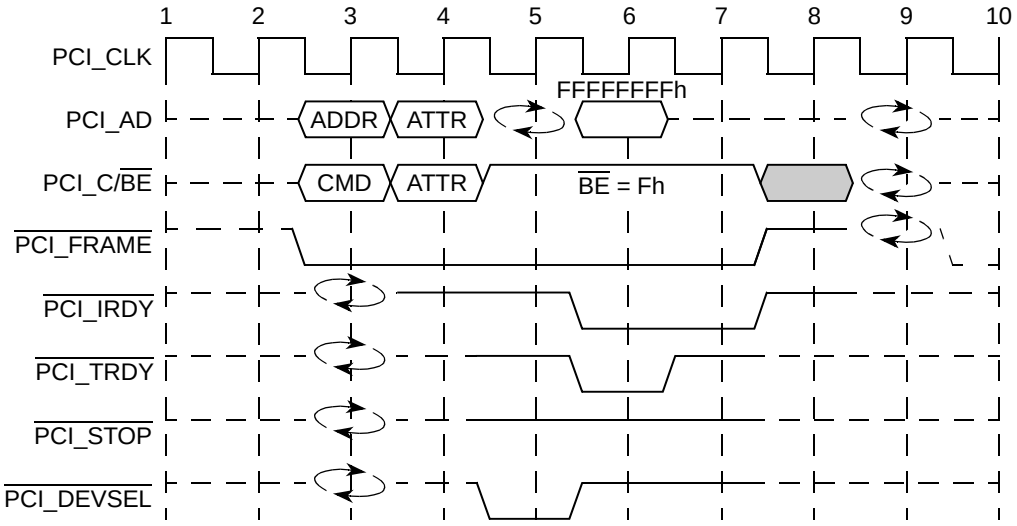


Figure 15-74. Split Response to a Read Transaction

Figure 15-74 shows a split response to a write transaction.

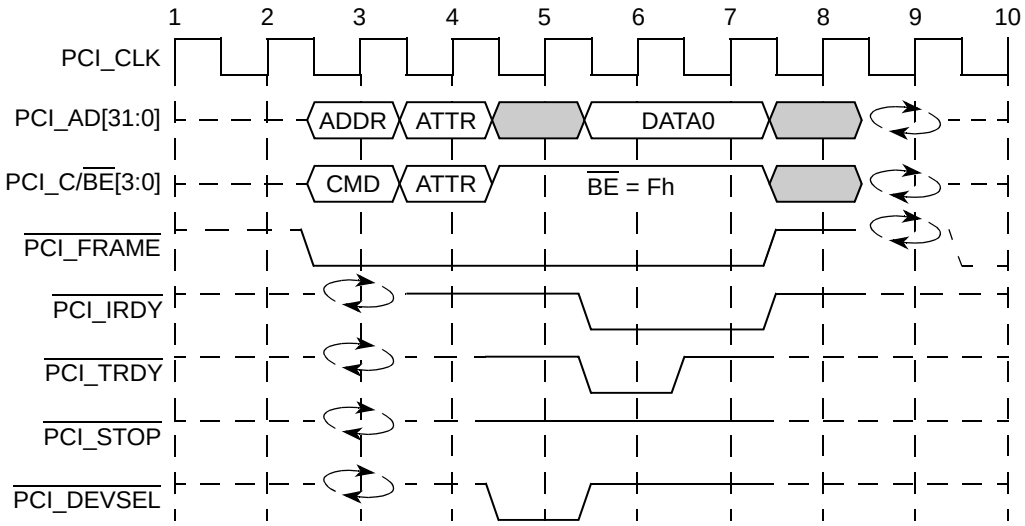


Figure 15-75. Split Response to a DWORD (4-Byte) Write Transaction

15.4.3.6.2 Completion Address

Figure 15-76 shows how AD[31:0] are driven during the address phase for split completion transactions.

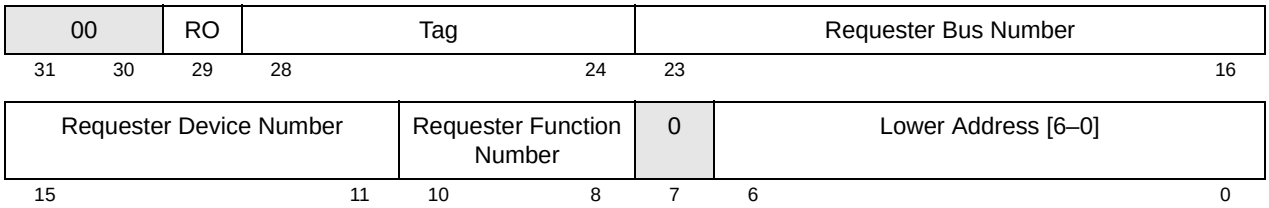


Figure 15-76. Split Completion Transaction Address AD[31:0]

Table 15-60 describes the fields of the attribute for Burst and DWORD transactions.

Table 15-60. Split Completion Transaction Address

AD	Name	Description
31:30	—	Reserved. Must be 00
30	RO	Relaxed ordering. The completer copies this field from the RO field of the requester attributes. Bridges optionally use this to influence transaction ordering. 0 The requester requires that the transaction remain in strict order. 1 The requester guarantees that the transaction is not required to remain in strict order.
28:24	Tag	The completer copies this field from the tag field of the requester attributes. The requester uses this information to identify the appropriate split completions.
23:16	Requester Bus Number	The completer copies this field from the requester bus number field of the requester attributes. The requester uses this information to identify the appropriate split completions.
15:11	Requester Device Number	The completer copies this field from the requester device number field of the requester attributes. The requester uses this information to identify the appropriate split completions.
10:8	Requester Function Number	The completer copies this field from the requester function number field of the requester attributes. The requester uses this information to identify the appropriate split completions.
7	—	Reserved. Must be 0
6:0	Lower Address	The completer copies the 7 lsbs of the split request address if all of the following are true: • The split request for this sequence is a burst read • This is the first split completion of the sequence • The split completion is not a split completion message. If the split completion is disconnected on an ADB, this field is cleared when the sequence resumes. If the split request is a DWORD transaction or the split completion is a split completion message, this field is cleared.

15.4.3.6.3 Completer Attributes

PCI-X split completion transactions include an attribute phase one clock cycle after the address phase. The completer attribute for a split completion transaction is similar to the burst transaction attribute. For the split completion attribute, PCI_C/$\overline{BE}$[3:0] provide the four msbs of the byte count and AD[31:0] are driven as shown in Figure 15-77.

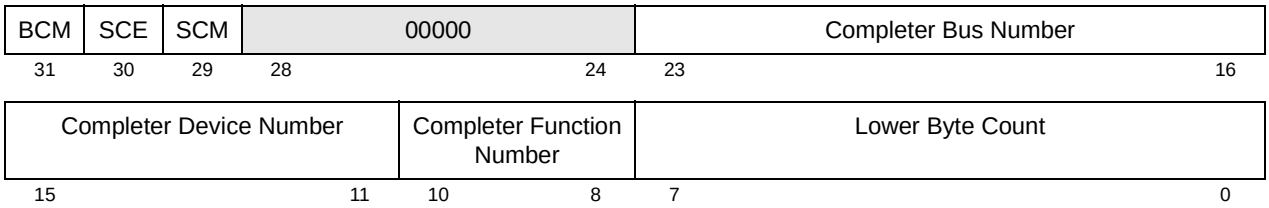


Figure 15-77. AD[31:0] During PCI-X Split Completion Attribute Phase

Table 15-61 describes the fields of the completer attribute for PCI-X split completion transactions.

Table 15-61. PCI-X Split Completion Transaction Attribute Summary

AD	Name	Description
31	BCM	Byte count modified. The completer must set BCM if the byte count is less than the full remaining byte count of the transaction. If the byte count is the full byte count of the split request, the completer clears BCM.
30	SCE	Split completion error. The completer sets this bit to indicate the transaction is a split completion message that includes an error message.
29	SCM	Split completion message. The completer clears this bit if the split completion contains read data. The completer sets this bit if the split completion contains a split completion message.
28:24	—	Reserved. Must be 00000
23:16	Completer Bus Number	The completer bus number attribute is supplied by the bus number field in the PCI-X status register.
15:11	Completer Device Number	The completer device number attribute is supplied by the device number field in the PCI-X status register.
10:8	Completer Function Number	The completer function number is assigned to the function in the completer by design.
7:0	Lower Byte Count	For a split completion attribute phase, PCI_C/$\overline{BE}$[3:0] is concatenated with AD[7:0] to create a 12-bit byte count. The byte count indicates the number of bytes remaining in the sequence.

15.4.3.6.4 Split Completion Messages

For PCI-X split completion transactions that include a split completion error or split completion message, the transaction has a single DWORD data phase with the format shown in Figure 15-78.

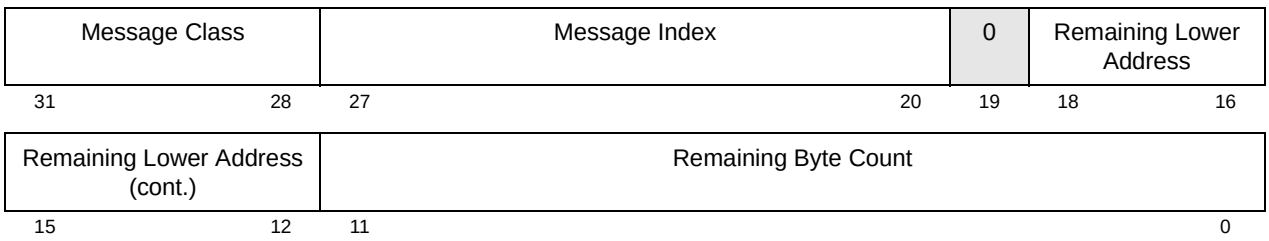


Figure 15-78. PCI-X Split Completion Message Format

Table 15-62 describes the fields of the PCI-X split completion message.

Table 15-62. PCI-X Split Completion Message Summary

AD	Name	Description
31:28	Message Class	A split completion message can be one of the following classes: 0x0 Write completion 0x1 PCI-X bridge error 0x2 Completer error 0x3–0xF Reserved
27:20	Message Index	The message index identifies the type of message within the message class. For write completion message class: 0x00 Normal completion For PCI-X bridge error message class: 0x00 Master-abort 0x01 Target-abort 0x02 Write data parity error For completer error message class: 0x00 Byte count out-of-range error 0x01 Split write data parity error 0x8n Device-specific error
19	—	Reserved
18:12	Remaining Lower Address	If the split request was a burst memory read transaction, the remaining lower address is the least significant seven bit of the first byte of data that has not been sent. If the split request was a DWORD transaction, the completer places all zeroes in this field.[1]
11:0	Remaining Byte Count	If the split request was a burst memory read transaction, the remaining byte count is the number of bytes that have not been sent. If the split request was a DWORD transaction, the completer places 0x004 in this field.

[1] Note that the PCI-X 1.0 specification states that if the split request is a DWORD transaction, the completer sets the remaining lower address field (bits 18–12) of the split completion message to zero. However, the MPC8560 deviates from the specification when the split request is a DWORD read transaction to the upper 32-bits of a quad word (that is, the address ends with 1xx); in this case, bit 14 of the split completion message is set to 1.

15.4.3.7 PCI-X Configuration Transactions

PCI-X configuration transactions are similar to PCI configuration transaction as described in Section 15.4.2.11, “Configuration Cycles.” The exceptions are noted in this section.

There is a rare situation where an internal access (originating from either the processor core, or other I/O ports on the device) to the 256-byte PCI-X configuration header may be corrupted by concurrent outbound PCI-X data transactions. The corruption does not occur if the PCI-X configuration transactions and the outbound PCI-X data transactions are not concurrent. The corruption does not occur with inbound PCI-X traffic. This is an issue for any system that mixes internally generated PCI-X configuration transactions with outbound PCI-X data transactions. There are two ways to avoid the situation:

1. Ensure that PCI-X configuration transactions are completed before starting PCI-X data transactions.
2. Ensure that all PCI-X outbound data traffic is quiesced before attempting PCI-X configuration transactions. The programmer must ensure all of the following:
   - Non-configuration transactions to PCI-X from the processor core are quiesced.
   - The core is not fetching instructions from PCI-X.
   - DMA transactions to PCI-X are quiesced.
   - RapidIO bridging transactions to PCI-X are quiesced.

As is the case for PCI configuration accesses, the bus master bit in the MPC8560's PCI bus command register must be set before an outbound configuration access is attempted. Figure 15-79 shows the PCI-X type 0 translation function performed on the contents of the PCI CFG_ADDR register to AD[31:0] on the PCI-X bus during the address phase of the configuration cycle.

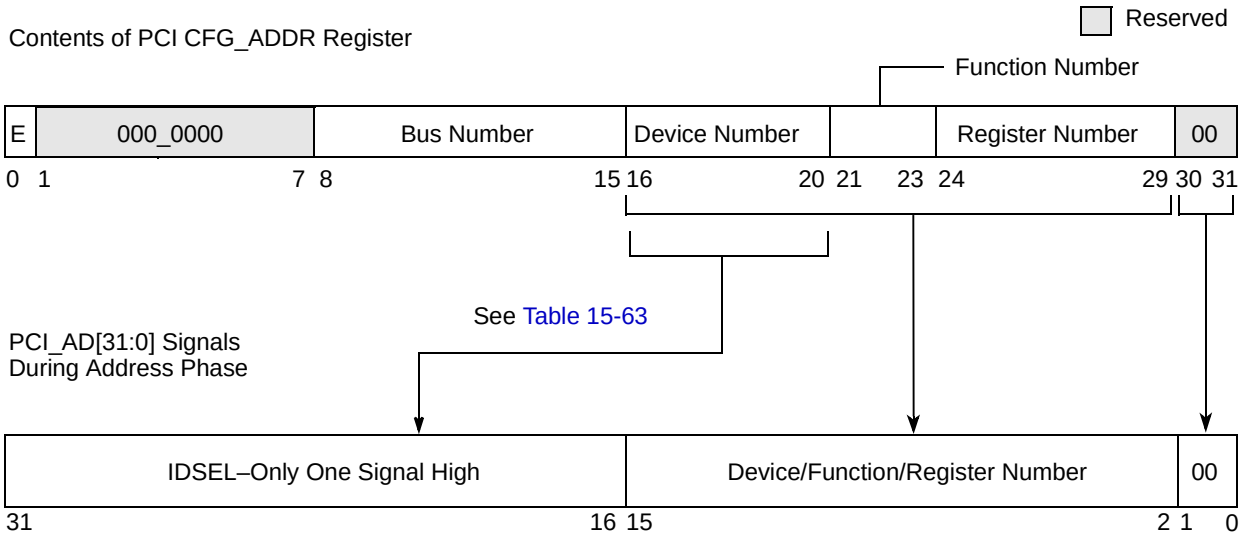


Figure 15-79. PCI-X Type 0 Configuration Translation

Compare the PCI-X translation in Figure 15-79 with the PCI configuration translation shown in Figure 15-66. Note that for PCI-X mode in addition to being used to generate the IDSEL signal, the device number is repeated on AD[15:11].

For PCI-X type 0 configuration cycles, the MPC8560 translates the device number field of PCI CFG_ADDR into a unique IDSEL signal for up to 15 different devices. Each device connects its IDSEL input to one of the AD[31:17] signals. For PCI-X type 0 configuration cycles, the MPC8560 translates the device number to AD*n* as shown in Table 15-63.

Table 15-63. PCI-X Type 0 Configuration—Device Number to AD*n* Translation

Device Number		AD Signal Used for IDSEL	Device Number		AD Signal Used for IDSEL
Binary	Decimal		Binary	Decimal	
0b0_0000	0	—	0b0_1001	9	AD25
0b0_0001	1	AD17	0b0_1010	10	AD26
0b0_0010	2	AD18	0b0_1011	11	AD27
0b0_0011	3	AD19	0b0_1100	12	AD28
0b0_0100	4	AD20	0b0_1101	13	AD29
0b0_0101	5	AD21	0b0_1110	14	AD30
0b0_0110	6	AD22	0b0_1111	15	AD31
0b0_0111	7	AD23	0b1_xxxx	—	—
0b0_1000	8	AD24	0b1_1111 [1]	31	—

[1] A device number of all ones indicates a PCI-X special-cycle or interrupt-acknowledge transaction.

For PCI-X type 0 translations, the function number and register number fields are copied without modification onto PCI_AD[15:2] during the address phase. PCI_AD[1:0] are driven to 0b00 during the address phase for PCI-X type 0 configuration cycles. The MPC8560 implements address stepping on configuration cycles so that the target's 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 PCI_AD[31:0] and PCI_C/$\overline{\text{BE}}$[3:0] four clocks before the assertion of $\overline{\text{PCI\_FRAME}}$.

PCI-X configuration transactions include an attribute phase one clock cycle after the address phase. The the attribute for a configuration transaction is similar to the DWORD transaction attribute. For the configuration attribute, PCI_C/$\overline{\text{BE}}$[3:0] provide the byte enables and AD[31:0] are driven as shown in Figure 15-80.

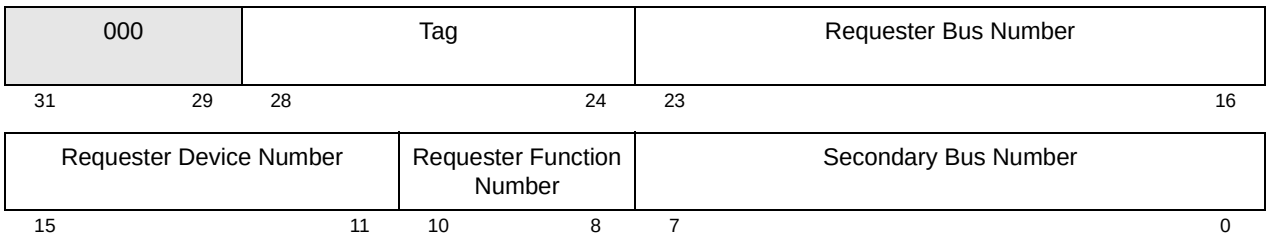


Figure 15-80. AD[31:0] During PCI-X Configuration Attribute Phase

Table 15-64 describes the fields of the attribute for PCI-X configuration transactions.

Table 15-64. PCI-X Configuration Transaction Attribute Summary

AD	Name	Description
31:29	—	Reserved. Must be 0
28:24	Tag	Identifies up to 32 unique sequences from a single initiator. Tag is also a component of the sequence ID.
23:16	Requester Bus Number	Supplied by the bus number field in the PCI-X status register. The requester bus number is also a component of the sequence ID.
15:11	Requester Device Number	Supplied by the device number field in the PCI-X status register. The requester bus number is also a component of the sequence ID.
10:8	Requester Function Number	Supplied by the function number field in the PCI-X status register. The requester bus number is also a component of the sequence ID.
7:0	Secondary Bus Number	The number of the bus on which the Type 0 configuration transaction is executing

15.4.3.8 PCI-X Error Functions

PCI-X provides for parity and other system errors to be detected and reported. This section describes generation and detection of parity and error reporting for the PCI-X bus.

15.4.3.8.1 Error Reporting

PCI-X 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 $\overline{\text{PCI\_SERR}}$. $\overline{\text{PCI\_PERR}}$ 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.

Note that some errors are enabled by programming two bits—one in the PCI bus command register and another in the PCI/X error enable register (ERR_EN). Likewise, some errors are reported in two bits—one in the PCI bus status register and another in the PCI/X error detect register (ERR_DR). These bits must be cleared separately; that is, clearing one does not clear the other. For example, clearing the ERR_DR[Mstr abort error] does not clear the received master abort bit in the PCI bus status register. In these cases, both bits must be cleared before further error reporting can occur.

Table 15-65 shows actions taken for each kind of error.

Table 15-65. PCI-X Mode Error Actions

PCI-X Error Type	Error Detect Register Bit	PCI Bus Status Register Bit	PCI-X Status Register Bit	Comment
PCI-X Outbound Read				
Received $\overline{\text{SERR}}$ at any phase	Rcvd $\overline{\text{SERR}}$ error	—	—	No data transferred
Received data parity error for data phase	Mstr $\overline{\text{PERR}}$ error	Detected parity error, Master data parity error detected	—	No data transferred
Master abort	Mstr abort error	Received master abort	—	No data transferred
Target abort	Trgt abort error	Received target abort	—	No data transferred
Memory space violation	ORMSV error	—	—	No data transferred. Only 8 bytes are requested in PCI bus
SC (split completion) with byte count value different than requested	SCM error	Signaled target abort	USC	No data transferred. Target abort
SC with wrong lower address field (burst read)	SCM error	Signaled target abort	USC	No data transferred. Target abort
SC with wrong lower address field (dword read)	—	—	—	
Extra SC. No pending SC with that tag	SCM error	—	USC	Master abort
SC error message	SCM error	—	RSM	No data transferred
SC not received before SC PCI-X timer time-out	TOE error	—	—	No data transferred
Detected parity error at address or attribute phase of SC	Addr parity error	Detected parity error, Signaled system error	—	No data transferred
Detected parity error at data phase of SC	Mstr $\overline{\text{PERR}}$ error	Detected parity error, Master data parity error detected	—	No data transferred
Received $\overline{\text{SERR}}$ at any phase of SC	Rcvd $\overline{\text{SERR}}$ error	—	—	No data transferred
PCI-X Outbound Write				
Received $\overline{\text{SERR}}$ at any phase	Rcvd $\overline{\text{SERR}}$ error	—	—	Float AD bus
Received $\overline{\text{PERR}}$ (data phase)	Rcvd $\overline{\text{PERR}}$ error	Master data parity error	—	
Master abort	Rcvd abort error	Received master abort	—	

Table 15-65. PCI-X Mode Error Actions (continued)

PCI-X Error Type	Error Detect Register Bit	PCI Bus Status Register Bit	PCI-X Status Register Bit	Comment
Target abort	Trgt abort error	Received target abort	—	
Memory space violation	OWMSV error	—	—	No data transferred. Only 8 bytes are requested in PCI bus
PCI-X Inbound Read				
Detected parity error for address or attribute phases	Addr parity error	Detected parity error, Signaled system error	—	Split completion error message generated
Detected parity error on upper address bits for address (SAC) or attribute phases	—	—	—	No PAR64 checks for SAC address or attribute phases
Detected parity error for address (DAC) phases	Addr parity error	Detected parity error, Signaled system error	—	Split completion error message generated
Received $\overline{\text{SERR}}$ at any phase	Rcvd $\overline{\text{SERR}}$ error	—	—	Split completion error message generated
Internal error	—	—	—	Split completion error message generated
Memory space violation	IRMSV error	—	—	Split completion error message generated
Received $\overline{\text{SERR}}$ at any phase of SC	Rcvd $\overline{\text{SERR}}$ error	—	—	Float AD bus
Received $\overline{\text{PERR}}$ at SC (data phase)	Trgt $\overline{\text{PERR}}$ error	—	—	
Master abort for SC	—	—	—	
Target abort for SC	—	—	—	
PCI-X Inbound Write				
Detected parity error for address or attribute phases	Addr parity error	Detected parity error, Signaled system error	—	Transaction purged
Detected parity error on upper address bits for address (SAC) or attribute phases	—	—	—	No PAR64 checks for SAC address or attribute phases
Detected parity error on upper address bits for address (DAC) phases	Addr parity error	Detected parity error, Signaled system error	—	Transaction purged
Received $\overline{\text{SERR}}$ at any phase	Rcvd $\overline{\text{SERR}}$ error	—	—	Transaction purged
Detected parity error for data phase	Trgt $\overline{\text{PERR}}$ error	Detected parity error	—	Transaction from the point of the error is purged

When an outbound PCI-X read transaction has been split and the split completion transaction indicates a master-abort or a target-abort, the split-completion error message (SCEM) data field contains information for setting the received master-abort or received target-abort bit in the PCI bus status register (bits 13 and 12, respectively). However, for misaligned reads (that is, reads not aligned on a 64-bit boundary), the error is not reflected in the PCI bus status register. The error is reported in the PCI error detection register (ERR_DR) at 0x0_8E00. The master-abort or target-abort can be inferred from the error data capture register (ERR_DL). Drivers that require this information can read the ERR_DR register rather than the PCI bus status register.

15.5 Initialization/Application Information

This section describes some tips for use of the PCI/X controller.

15.5.1 Power-On Reset Configuration Modes

The PCI/X block can power-on in three modes: host mode, agent mode and agent configuration 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 Freescale Semiconductor parts and PCI implementations. Note that after reset, the device cannot be switched from one mode to another.

The affected configuration bits are defined in Table 15-66.

Table 15-66. Affected Configuration Register Bits for POR

Register (offset)	Bit	Name	Register Description
PCI bus command register (0x04)	2	Bus master	Controls whether the device can master a transaction on the PCI/X bus. If cleared, the device can not master a transaction. This bit is independent of host or agent mode.
	1	Memory space	Controls acknowledgement of inbound memory transactions. If cleared, all inbound memory accesses (including accesses to PCSRBAR space) end in a master abort. This bit is independent of host or agent mode.
PCI bus function register (0x44)	5	ACL	Valid only in agent mode. Controls 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.
	0	PAH	Determines whether the device is in agent or host mode. Zero indicates host mode.

The POR reset values for the affected configuration bits are described in Table 15-67.

Table 15-67. Power-On Reset Values for Affected Configuration Bits

Mode	Configuration Bit			
	Bus Master	Memory Space	ACL	PAH
Host	1	0	X	0
Agent	0	0	0	1
Agent configuration lock	0	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 device powers up with the ability to master transactions on the PCI bus, however in order to acknowledge memory transactions, the memory space bit must be set.

15.5.1.2 Agent Mode

When the device powers up in agent mode, it acknowledges inbound configuration accesses. However the device cannot master transactions or acknowledge inbound memory accesses on the PCI bus until the appropriate configuration bits (bus master and memory space, respectively) have been set.

15.5.1.3 Agent Configuration Lock Mode

Agent configuration lock mode is similar to agent mode with the added restriction that when the device powers up in agent configuration lock mode, it retries all inbound configuration accesses until the ACL bit is cleared. As in agent mode, the device in agent configuration lock mode cannot master transactions or acknowledge inbound memory accesses on the PCI bus until the appropriate configuration bits (bus master and memory space, respectively) have been set.

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. To fully serialize these non-posted writes, the processor should follow each write transaction with a read of the same target, followed by proper barrier instructions to guarantee completion of the read before continuing with a subsequent write. This method guarantees completion of the write and serialization of a series of writes.

15.5.3 PCI-X Outbound Read Transaction Alignment

The PCI-X protocol supports only read transactions with byte enables that are 32-bit aligned and are 32 bits wide. The MPC8560 requires the following:

1. Reads to PCI-X greater than 32 bits must be aligned to a 64-bit boundary and have a size that is a multiple of 8 bytes.
2. Reads to PCI-X less than or equal to 32 bits must not cross a 32-bit boundary.

A read to PCI-X that is not aligned to a 32-bit boundary that crosses a 32-bit boundary will execute only to the 32-bit boundary. No mechanism is provided in hardware to split this type of read into multiple transactions with byte enables on the PCI-X bus.

Care should be taken on the MPC8560 when writing software to meet these alignment restrictions.

15.5.4 PCI-X Inbound Reads that Cross Window Boundaries

For Rev 1 of the MPC8560, inbound PCI-X read requests can cross a window boundary. For Rev 2, if an inbound PCI-X read request crosses a window boundary, the MPC8560 responds with a split completion message indicating an error but does not return any data. This is a deviation from the PCI-X specification which describes that data is returned up to the window boundary.

15.5.5 Bridging Between RapidIO and PCI/X

The MPC8560 does not support bridging from RapidIO according to the RapidIO Interoperability Specification. If an external RapidIO device tries to target the MPC8560 with a RapidIO NWRITE-R transaction or with any RapidIO write at a priority level higher than 0, and the RapidIO inbound ATMU routes the transaction to the PCI/PCI-X interface, the MPC8560 could deadlock internally.

Therefore, the use of RapidIO NWRITE-R transactions must be avoided in bridging applications to the PCI/X controller, and all RapidIO writes targeting the PCI/X controller must be of priority level 0. This means transaction ordering is relaxed because read responses will not necessarily push posted writes in the bridge, and writes will never bypass read requests. Note that this specifically precludes the topology where two MPC8560s communicating with each other via PCI or PCI-X may deadlock when two RapidIO devices (one on each MPC8560) are simultaneously performing priority-1 writes to targets on the remote MPC8560 through the PCI/X link between the devices. To support such a topology, RapidIO write transactions that bridge to the other MPC8560 must be priority 0.

Chapter 16
RapidIO Interface

The RapidIO controller conforms to Revision 1.2 of the *Parallel RapidIO Interconnect Specification*. This chapter describes the implementation of the parallel RapidIO controller.

16.1 Introduction

The RapidIO controller is partitioned into inbound and outbound blocks, and these blocks are further partitioned into three implementation layers: physical, protocol, and logical. Note that these implementation layers do not precisely correlate with the layers described in the *Parallel RapidIO Interconnect Specification*. The physical and protocol layers primarily address the portion of the *Parallel RapidIO Interconnect Specification* titled “Physical Layer 8/16 LP-EP.” The logical layer refers primarily to the logical portions of the *Parallel 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 device frequency.

16.1.1 Overview

A block diagram of The RapidIO controller is shown in Figure 16-1 and includes a RapidIO interface and a system core interface; those signals are divided into several groups. These groups include: the RapidIO request, response, ack done, and data transfer interfaces, the system core 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.

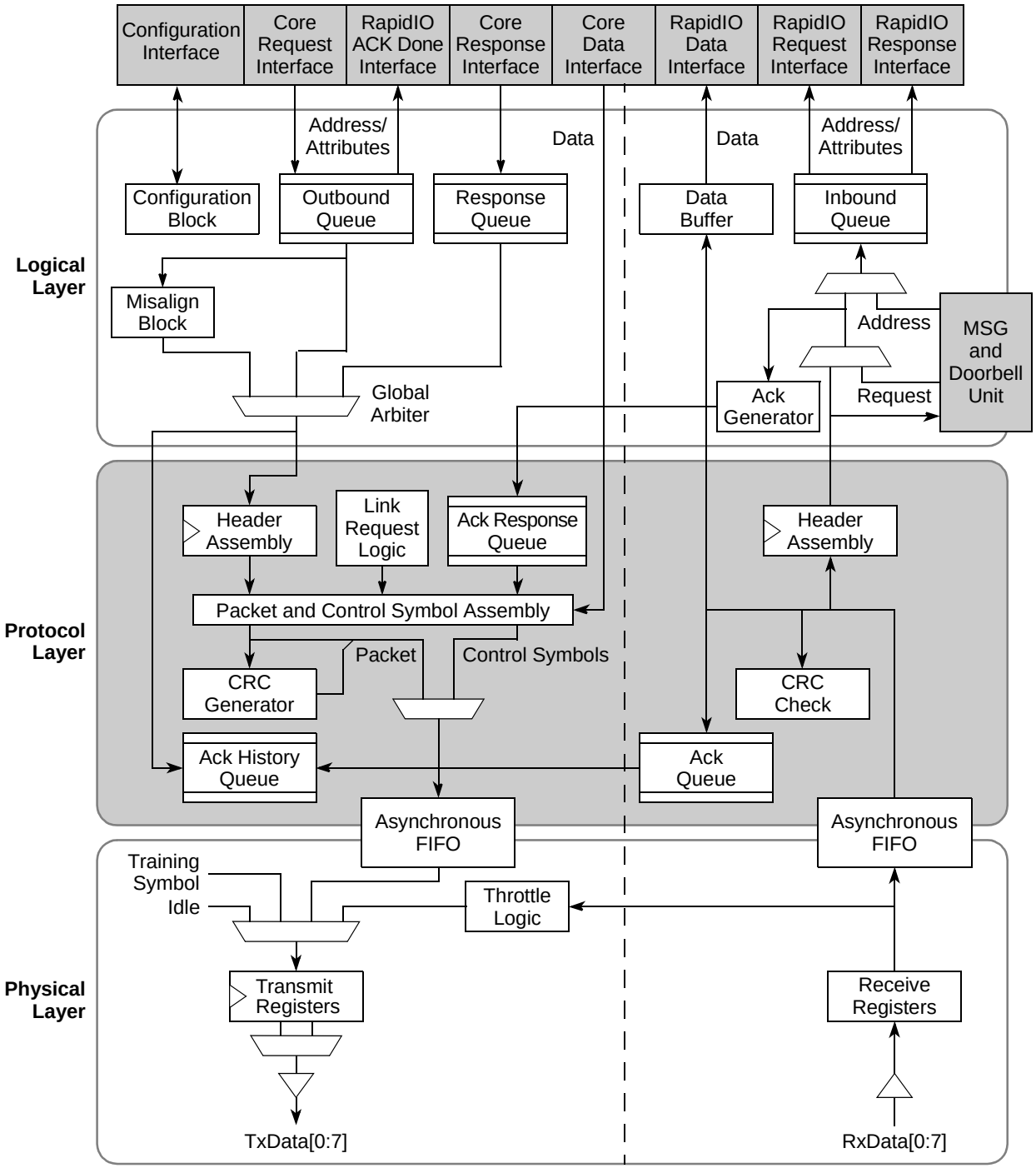


Figure 16-1. RapidIO High Level Block Diagram

The RapidIO controller also implements extra process-specific circuitry for receive clock recovery and for transmit clock and data alignment.

The RapidIO inbound interface receives and latches 8 double data rate (DDR) data bits. 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. After the packet header information arrives in the inbound packet assembly logic, packets are stripped of cyclical redundancy check (CRC) information and other transport information not required by the logical layer. The 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 the RapidIO controller block and into the device. 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 device 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 device 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 Descriptions,” describes the interface signal groups, their meanings, and timing relationships.
- Section 16.3, “Memory Map/Register Definition,” describes the configuration, runtime, 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,” provides general information about implementing RapidIO features.

16.1.2 RapidIO Terminology

Table 16-1 describes terminology used throughout this chapter.

Table 16-1. RapidIO Terminology

Term	Description
Ack	Acknowledge control symbol
CAR	Capability register
CRC	Cyclic redundancy check
CSR	Command and status register
Domain	Logically associated group of processing elements
Doorbell	Port on a device that generates an interrupt to the processor
Double word	An 8-byte quantity
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 between the protocol layer and the physical layer
LP-LVDS	Low-power, low-voltage differential signaling
Mailbox	Dedicated hardware that receives a message in the message passing unit
Physical layer	Physical handling of the transport of packets over the RapidIO interface
Priority	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	Handling of packets at the protocol level on the RapidIO interface
Receiver	Destination of a packet on the RapidIO interface, also referred to as a destination or a target
Source	Originator of a packet on the RapidIO interconnect
Symbol	One beat of a packet appearing on the interface pins
Target	Destination of a packet on the RapidIO interface, also referred to as a destination or a receiver
Word	A 4-byte quantity

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 *Parallel RapidIO Interconnect 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
- Default and configurable inbound ATMU window
- Default outbound ATMU window
- CRC-based error detection (CCITT-CRC16) as described in the 8/16 LP-EP physical specification
- 256-byte data payloads
- Packet pacing/retry capability
- Only the small size transport information field (TT = 0b00)
- Link validation, error recovery, training, and 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 *Parallel RapidIO Intconnect Specification* are not supported:

- Large size transport information field (TT = 0b01)
- TOD-sync control symbols (these are treated as idle control symbols)
- Atomic swap or 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 signal. The clock source is selected at power-on reset, as described in Section 4.4.3.10, “RapidIO Transmit Clock Source,” and Section 4.4.4.2, “RapidIO Clocks.” Note that the CCB clock is the clock signal for the rest of the RapidIO block.

16.1.4.2 CRC Checking Modes

RapidIO offers the modes for packet CRC checking shown in Table 16-2. Note that a read that attempts to access an unmapped target causes the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, PCR[CCE] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

Table 16-2. CRC Checking Mode

PCR[CCE]	Mode
0	CRC checking disabled
1	CRC enabled with hardware-based recovery

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, controlled by PCCSR[OPD], disables RapidIO input receivers and output drivers.

16.1.4.6 Accept All Mode

This mode is controlled by a bit (CR[AA]) in the RapidIO configuration register. While this mode is enabled, all packets are accepted regardless of the target ID. Table 16-3 describes the behavior with respect to the accept all mode.

Table 16-3. Accept All Mode Behavior

Accept All (CR[AA])	Device Behavior for Target ID ≠ Device ID
0	Error reported in PNFEDR[ITE].
1	Accept and process the packet without regard to the target ID

16.1.4.7 Packet Response Time-Out Disable Mode

The RapidIO packet response time-out counters can be disabled by setting a bit in the RapidIO port notification/fatal error disable register. For more information, see Section 16.3.2.3.2, “Port Notification/Fatal Error Detect Disable Register (PNFEDiR).”

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 port notification/fatal error disable register. For more information, see Section 16.3.2.3.2, “Port Notification/Fatal Error Detect Disable Register (PNFEDiR).”

16.2 External Signal Descriptions

This section describes the signal names of the RapidIO block, their meanings, and the timing relationships that exist among the signals.

16.2.1 Signals Overview

The RapidIO block implements the functionality described in the *Parallel RapidIO Intconnect Specification* for an 8-bit parallel interface with separate transmit and receive ports as shown in Figure 16-2.

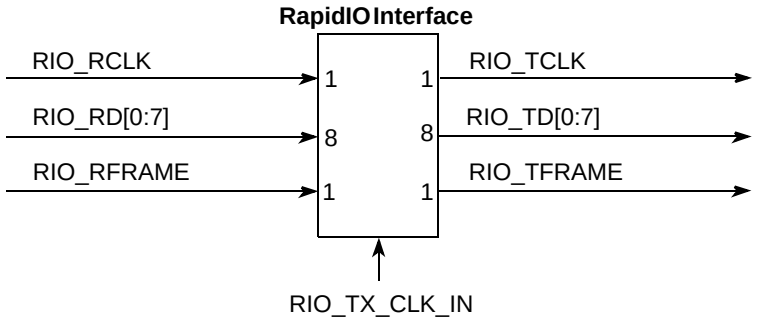


Figure 16-2. RapidIO Interface Signals Model

The RapidIO signals are implemented as differential pairs of signals. Table 16-4 lists the external RapidIO signals implemented on this interface.

Table 16-4. RapidIO Interface Signals Summary

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 16-5 describes the RapidIO interface signals in detail.

Table 16-5. RapidIO Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
RIO_RCLK	I	Receive clock. Free-running input clock used to capture control and data signals. RIO_RCLK is used to capture the frame signal RIO_RFRAME and the data bus RIO_RD[0:7]. RIO_RCLK is the differential pair of RIO_RCLK.	
		State Meaning	Asserted/Negated—A clock event is determined as the rising or falling edge of RIO_RCLK.
		Timing	Assertion/Negation—Free-running clock. See the *MPC8560 Integrated Processor 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 RIO_RCLK. RIO_RD[0:7] is the differential pair of RIO_RD[0:7]	
		State Meaning	Asserted/Negated—Represents the data being received on the RapidIO interface.
		Timing	RIO_RD[0:7] data is captured on both the rising and falling edges of RIO_RCLK.
RIO_RFRAME	I	Receive frame. Non return to zero (NRZ) input. It toggles at the beginning of a packet or control symbol. RIO_RFRAME is the differential pair of RIO_RFRAME.	
		State Meaning	Asserted/Negated—Assertion is signalled by a toggling of the signal. Toggling indicates a special packet framing event on the receive data (RIO_RD) pins.
		Timing	RIO_RFRAME is sampled with respect to RIO_RCLK.
RIO_TCLK	O	Transmit clock out. Free-running output clock which is launched in phase with the output data, RIO_TD, and the frame signal, RIO_TFRAME. RIO_TCLK should track the data bus RIO_TD[0:7] through all the associated routing (board, package, and so forth). RIO_TCLK is the differential pair of RIO_TCLK. See Section 16.1.4.1, “Transmit Clock-Select Mode,” for more information on the source of the transmit clock.	
		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 device clock.
RIO_TD[0:7]	O	Transmit data. Carries packet information as well as control symbols. The data bus is switched on both edges of RIO_TCLK. RIO_TD[0:7] is the differential pair of RIO_TD[0:7].	
		State Meaning	Asserted/Negated—Represents the data being transmitted on the RapidIO interface.
		Timing	Assertion of RIO_TD is always performed with a fixed relationship to RIO_TCLK.

Table 16-5. RapidIO Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description
RIO_TFRAME	I	Transmit frame. NRZ output. It toggles at the beginning of a packet or control symbol. $\overline{\text{RIO\_TFRAME}}$ 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. Externally generated reference clock for the RapidIO transmit domain. $\overline{\text{RIO\_TX\_CLK\_IN}}$ is the differential pair of RIO_TX_CLK_IN.

16.3 Memory Map/Register Definition

The memory map for the RapidIO controller is shown in Table 16-6.

Table 16-6. RapidIO Module Memory Map

Offset	Register	Access	Reset	Section/Page
RapidIO Architectural Registers				
0xC_0000	DIDCAR—Device identity capability register	R	0x0003_0002	16.3.1.1/16-13
0xC_0004	DICAR—Device information capability register	R	0x8070_0020	16.3.1.2/16-14
0xC_0008	AIDCAR—Assembly identity capability register	R/W	0x0000_0000	16.3.1.3/16-14
0xC_000C	AICAR—Assembly information capability register	R/W	0x0000_0100	16.3.1.4/16-15
0xC_0010	PEFCAR—Processing element features capability register	R	0xE088_0009	16.3.1.5/16-15
0xC_0014	SPICAR—Switch port information capability register	R	0x0000_0100	16.3.1.6/16-17
0xC_0018	SOCAR—Source operations capability register	R	0x0600_FCF0	16.3.1.7/16-17
0xC_001C	DOCAR—Destination operations capability register	R	0x0600_FCF4	16.3.1.8/16-19
0xC_0040	MSR—Mailbox command and status register	R	0x0000_0000	16.3.1.9/16-22
0xC_0044	PWDCSR—Port-write and doorbell command and status register	R	0x0000_0020	16.3.1.10/16-22
0xC_004C	PELLCCSR—Processing element logical layer control command and status register	R	0x0000_0001	16.3.1.11/16-24
0xC_0058	Reserved	R	0x0000_0000	—
0xC_005C	LCSBA1CSR—Local configuration space base address 1 command and status register	R/W	0x0000_0000	16.3.1.12/16-24
0xC_0060	BDIDCSR—Base device ID command and status register	R/W	0x00*nn*_0000	16.3.1.13/16-25
0xC_0068	HBDIDLCSR—Host base device ID lock command and status register	R/W	0x0000_FFFF	16.3.1.14/16-26
0xC_006C	CTCSR—Component tag command and status register	R/W	0x0000_0000	16.3.1.15/16-26
0xC_0100	PMBH0CSR—8/16 LP-LVDS port maintenance block header 0 command and status register	R	0x0000_0002	16.3.1.16/16-27

Table 16-6. RapidIO Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xC_0120	PLTOCCSR—Port link time-out control command and status register	R/W	0xFFFF_FF00	16.3.1.17/16-27
0xC_0124	PRTOCCSR—Port response time-out control command and status register	R/W	0xFFFF_FF00	16.3.1.18/16-28
0xC_013C	PGCCSR—Port general control command and status register	R/W	0x*n*000_0000	16.3.1.19/16-29
0xC_0140	PLMREQCSR—Port link maintenance request command and status register	R/W	0x0000_0000	16.3.1.20/16-29
0xC_0144	PLMRESPCSR—Port link maintenance response command and status register	R	0x0000_0000	16.3.1.21/16-30
0xC_0148	PLASCSR—Port local ackID status command and status register	R/W	0x0000_0000	16.3.1.22/16-31
0xC_0158	PESCSR—Port error and status command and status register	R/W	0x0000_0000	16.3.1.23/16-32
0xC_015C	PCCSR—Port control command and status register	R/W	0x4400_0000	16.3.1.24/16-33
RapidIO Implementation Registers				
0xD_0000	CR—Configuration register	R/W	0x0000_0001	16.3.2.1.1/16-34
0xD_0010	PCR—Port configuration register	R/W	0x0000_0010	16.3.2.1.2/16-35
0xD_0014	PEIR—Port error injection register	R/W	0x0000_0000	16.3.2.1.3/16-35
RapidIO ATMU Registers				
0xD_0C00	ROWTAR0—RapidIO outbound window translation address register 0	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0C10	ROWAR0—RapidIO outbound window attributes register 0	R/W	0x8004_401F	16.3.2.2.3/16-39
0xD_0C20	ROWTAR1—RapidIO outbound window translation address register 1	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0C28	ROWBAR1—RapidIO outbound window base address register 1	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0C30	ROWAR1—RapidIO outbound window attributes register 1	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0C40	ROWTAR2—RapidIO outbound window translation address register 2	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0C48	ROWBAR2—RapidIO outbound window base address register 2	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0C50	ROWAR2—RapidIO outbound window attributes register 2	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0C60	ROWTAR3—RapidIO outbound window translation address register 3	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0C68	ROWBAR3—RapidIO outbound window base address register 3	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0C70	ROWAR3—RapidIO outbound window attributes register 3	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0C80	ROWTAR4—RapidIO outbound window translation address register 4	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0C88	ROWBAR4—RapidIO outbound window base address register 4	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0C90	ROWAR4—RapidIO outbound window attributes register 4	R/W	0x0004_401F	16.3.2.2.3/16-39

Table 16-6. RapidIO Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_0CA0	ROWTAR5—RapidIO outbound window translation address register 5	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0CA8	ROWBAR5—RapidIO outbound window base address register 5	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0CB0	ROWAR5—RapidIO outbound window attributes register 5	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0CC0	ROWTAR6—RapidIO outbound window translation address register 6	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0CC8	ROWBAR6—RapidIO outbound window base address register 6	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0CD0	ROWAR6—RapidIO outbound window attributes register 6	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0CE0	ROWTAR7—RapidIO outbound window translation address register 7	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0CE8	ROWBAR7—RapidIO outbound window base address register 7	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0CF0	ROWAR7—RapidIO outbound window attributes register 7	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0D00	ROWTAR8—RapidIO outbound window translation address register 8	R/W	0x0000_0000	16.3.2.2.1/16-37
0xD_0D08	ROWBAR8—RapidIO outbound window base address register 8	R/W	0x0000_0000	16.3.2.2.2/16-38
0xD_0D10	ROWAR8—RapidIO outbound window attributes register 8	R/W	0x0004_401F	16.3.2.2.3/16-39
0xD_0D60	RIWTAR4—RapidIO inbound window translation address register 4	R/W	0x0000_0000	16.3.2.2.4/16-41
0xD_0D68	RIWBAR4—RapidIO inbound window base address register 4	R/W	0x0000_0000	16.3.2.2.5/16-41
0xD_0D70	RIWAR4—RapidIO inbound window attributes register 4	R/W	0x0004_401F	16.3.2.2.6/16-42
0xD_0D80	RIWTAR3—RapidIO inbound window translation address register 3	R/W	0x0000_0000	16.3.2.2.4/16-41
0xD_0D88	RIWBAR3—RapidIO inbound window base address register 3	R/W	0x0000_0000	16.3.2.2.5/16-41
0xD_0D90	RIWAR3—RapidIO inbound window attributes register 3	R/W	0x0004_401F	16.3.2.2.6/16-42
0xD_0DA0	RIWTAR2—RapidIO inbound window translation address register 2	R/W	0x0000_0000	16.3.2.2.4/16-41
0xD_0DA8	RIWBAR2—RapidIO inbound window base address register 2	R/W	0x0000_0000	16.3.2.2.5/16-41
0xD_0DB0	RIWAR2—RapidIO inbound window attributes register 2	R/W	0x0004_401F	16.3.2.2.6/16-42
0xD_0DC0	RIWTAR1—RapidIO inbound window translation address register 1	R/W	0x0000_0000	16.3.2.2.4/16-41
0xD_0DC8	RIWBAR1—RapidIO inbound window base address register 1	R/W	0x0000_0000	16.3.2.2.5/16-41
0xD_0DD0	RIWAR1—RapidIO inbound window attributes register 1	R/W	0x0004_401F	16.3.2.2.6/16-42
0xD_0DE0	RIWTAR0—RapidIO inbound window translation address register 0	R/W	0x0000_0000	16.3.2.2.4/16-41
0xD_0DF0	RIWAR0—RapidIO inbound window attributes register 0	R/W	0x8004_401F	16.3.2.2.6/16-42

Table 16-6. RapidIO Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
RapidIO Error Management Registers				
0xD_0E00	PNFEDR—Port notification/fatal error detect register	R/W	0x0000_0000	16.3.2.3.1/16-44
0xD_0E04	PNFEDiR—Port notification/fatal error detect disable register	R/W	0x0000_0000	16.3.2.3.2/16-47
0xD_0E08	PNFEIER—Port notification/fatal error interrupt enable register	R/W	0x0000_0000	16.3.2.3.3/16-49
0xD_0E0C	PECSR—Port error capture status register	R/W	0x0000_0000	16.3.2.3.4/16-52
0xD_0E10	EPCR0—Error packet capture register 0	R/W	0x0000_0000	16.3.2.3.5/16-52
0xD_0E14	EPCR1—Error packet capture register 1	R/W	0x0000_0000	16.3.2.3.6/16-53
0xD_0E18	EPCR2—Error packet capture register 2	R/W	0x0000_0000	16.3.2.3.16/16-58
0xD_0E20	PREDR—Port recoverable error detect register	R/W	0x0000_0000	16.3.2.3.23/16-61
0xD_0E28	PERTR—Port error recovery threshold register	R/W	0x00FF_0000	16.3.2.3.24/16-64
0xD_0E2C	PRTR—Port retry threshold register	R/W	0x00FF_0000	16.3.2.3.25/16-64
RapidIO Message Unit				
RapidIO Outbound Message Registers				
0xD_1000	OMR—Outbound mode register	R/W	0x0000_0000	16.3.3.1.1/16-65
0xD_1004	OSR—Outbound status register	R/W	0x0000_0000	16.3.3.1.2/16-67
0xD_100C	ODQDPAR—Outbound descriptor queue dequeue pointer address register	R/W	0x0000_0000	16.3.3.1.3/16-68
0xD_1014	OSAR—Outbound source address register	R/W	0x0000_0000	16.3.3.1.4/16-69
0xD_1018	ODPR—Outbound destination port register	R/W	0x0000_0000	16.3.3.1.5/16-70
0xD_101C	ODATR—Outbound destination attributes register	R/W	0x0006_0000	16.3.3.1.6/16-70
0xD_1020	ODCR—Outbound double-word count register	R/W	0x0000_0000	16.3.3.1.7/16-71
0xD_1028	ODQEPAR—Outbound descriptor queue enqueue pointer address register	R/W	0x0000_0000	16.3.3.1.8/16-72
RapidIO Inbound Message Registers				
0xD_1060	IMR—Inbound mailbox mode register	R/W	0x0000_0000	16.3.3.2.1/16-72
0xD_1064	ISR—Inbound mailbox status register	R/W	0x0000_0000	16.3.3.2.2/16-74
0xD_106C	IFQDPAR—Inbound frame queue dequeue pointer address register	R/W	0x0000_0000	16.3.3.2.3/16-75
0xD_1074	IFQEPAR—Inbound frame queue enqueue pointer address register	R/W	0x0000_0000	16.3.3.2.4/16-76
RapidIO Doorbell Registers				
0xD_1460	DMR—Doorbell mode register	R/W	0x0000_0000	16.3.3.3.1/16-77
0xD_1464	DSR—Doorbell status register	R/W	0x0000_0000	16.3.3.3.2/16-78

Table 16-6. RapidIO Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_146C	DQDPAR—Doorbell queue dequeue pointer address register	R/W	0x0000_0000	16.3.3.3.3/16-79
0xD_1474	DQEPAR—Doorbell queue enqueue pointer address register	R/W	0x0000_0000	16.3.3.3.4/16-80
RapidIO Port-Write Registers				
0xD_14E0	PWMR—Port-write mode register	R/W	0x0000_0000	16.3.3.4.1/16-81
0xD_14E4	PWSR—Port-write status register	R/W	0x0000_0000	16.3.3.4.2/16-82
0xD_14EC	PWQBAR—Port-write queue base address register	R/W	0x0000_0000	16.3.3.4.3/16-83

16.3.1 Architectural Registers

Following are the architectural registers of the RapidIO module. For additional details of these registers, refer to the *Parallel RapidIO Intconnect Specification*.

16.3.1.1 Device Identity Capability Register (DIDCAR)

The DIDCAR is a read-only register that provides information regarding the type of device being used as well as the vendor who manufactured it. Figure 16-3 describes the DIDCAR.

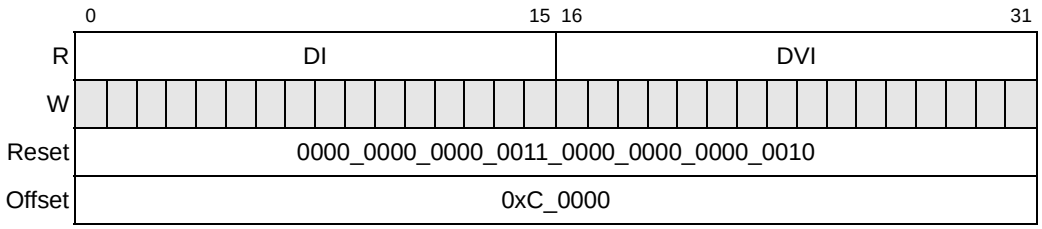


Figure 16-3. Device Identity Capability Register (DIDCAR)

Table 16-7 describes the fields of the DIDCAR.

Table 16-7. DIDCAR Field Descriptions

Bits	Name	Description
0–15	DI	Device identity field. Uniquely identifies the type of device from the vendor specified in the device vendor identity field. The values of the device identity field are assigned and managed by the vendor. 0x0003—MPC8560
16–31	DVI	Device vendor identity field. Identifies the vendor that manufactured the device. A value is uniquely assigned to a device vendor by the registration authority of the RapidIO Trade Association. 0x0002—Freescale

16.3.1.2 Device Information Capability Register (DICAR)

The DICAR, shown in Figure 16-4, identifies the revision level of the device. The value is assigned and managed by the vendor specified in DIDCAR[DVI]. DICAR represents a copy of the device's system version register (SVR).

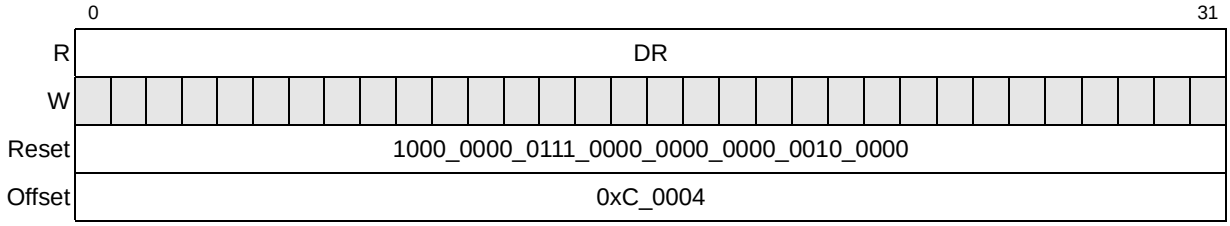


Figure 16-4. Device Information Capability Register (DICAR)

Table 16-8 describes the field of the DICAR.

Table 16-8. DICAR Field Description

Bits	Name	Description
0–31	DR	Device revision—Copy of the device's system version register (SVR). Dracom: 0x8070_0020

16.3.1.3 Assembly Identity Capability Register (AIDCAR)

AIDCAR, shown in Figure 16-5, provides information regarding the type of assembly or subsystem being used as well as the vendor who manufactured it.

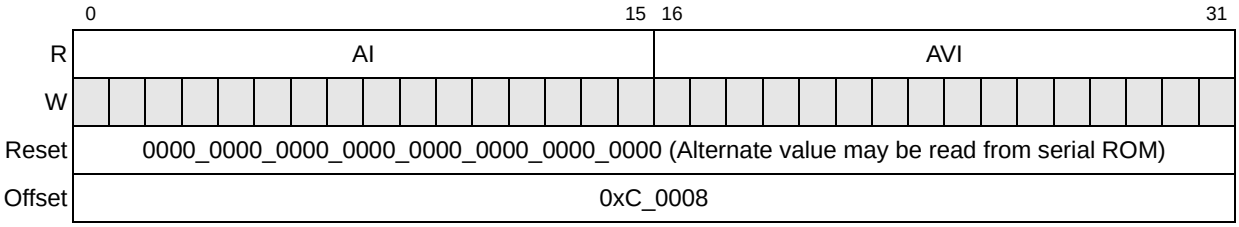


Figure 16-5. Assembly Identity Capability Register (AIDCAR)

Table 16-9 describes the fields of the AIDCAR.

Table 16-9. AIDCAR Field Descriptions

Bits	Name	Description
0–15	AI	Assembly identity field. The assembly field uniquely identifies the type of assembly or subsystem from the vendor specified in the assembly vendor identity field. The value of the assembly identity field by default is all zeros but may be assigned by a boot ROM device.
16–31	AVI	Assembly vendor identity field. The assembly vendor identity field identifies the vendor that manufactured the assembly or subsystem. A value for the assembly vendor identity field is uniquely assigned to an assembly vendor by the registration authority of the RapidIO Trade Association. By default, this value is all zeros but may be assigned by a boot ROM device.

16.3.1.4 Assembly Information Capability Register (AICAR)

The AICAR, shown in Figure 16-6, provides additional information about the assembly.

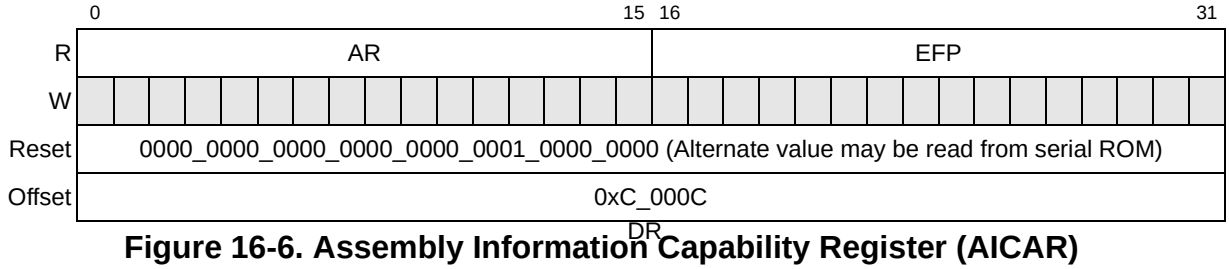


Figure 16-6. Assembly Information Capability Register (AICAR)

Table 16-10 describes the fields of the AICAR.

Table 16-10. AICAR Field Descriptions

Bits	Name	Description
0–15	AR	Assembly revision. This field states the assembly revision level.
16–31	EFP	Extended features pointer. Pointer to the first entry in the extended features list.

16.3.1.5 Processing Element Features Capability Register (PEFCAR)

The PEFCAR, shown in Figure 16-7, identifies major functionality provided by the RapidIO block.

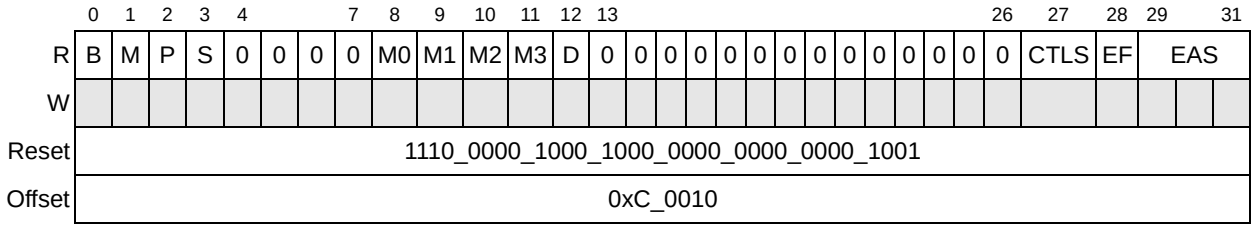


Figure 16-7. Processing Element Features Capability Register (PEFCAR)

Table 16-11 describes the fields of the PEFCAR.

Table 16-11. PEFCAR Field Descriptions

Bits	Name	Description
0	B	Bridge. Set for the MPC8560. 0 The RapidIO controller does not have a DRAM interface. 1 The RapidIO controller has a DRAM interface.
1	M	Memory. Set for the MPC8560. 0 The RapidIO controller does not have local address space. 1 The RapidIO controller has local address space.

Table 16-11. PEFCAR Field Descriptions

Bits	Name	Description
2	P	Processor. Set for the MPC8560. 0 The RapidIO controller does not have a processor core. 1 The RapidIO controller has a processor core.
3	S	Switch. Cleared for the MPC8560. 0 The RapidIO controller is a switch. 1 The RapidIO controller is not a switch.
4–7	—	Reserved
8	M0	Mailbox 0. Set for the MPC8560. 0 The RapidIO controller does not support mailbox 0. 1 The RapidIO controller supports mailbox 0.
9	M1	Mailbox 1. Cleared for the MPC8560. 0 The RapidIO controller does not support mailbox 1. 1 The RapidIO controller supports mailbox 1.
10	M2	Mailbox 2. Cleared for the MPC8560. 0 The RapidIO controller does not support mailbox 2. 1 The RapidIO controller supports mailbox 2.
11	M3	Mailbox 3. Cleared for the MPC8560. 0 The RapidIO controller does not support mailbox 3. 1 The RapidIO controller supports mailbox 3.
12	D	Doorbell. Set for the MPC8560. 0 The RapidIO controller does not support inbound doorbells. 1 The RapidIO controller supports inbound doorbells.
13–26	—	Reserved
27	CTLS	Common transport route. Cleared for the MPC8560. 0 The RapidIO controller does not support the large common transport route field. 1 The RapidIO controller supports the large common transport route field.
28	EF	Extended features. Set for the MPC8560. 0 The extended features pointer is not valid. 1 The extended features pointer is valid.
29–31	EAS	Extended addressing support. Possesses a value of 001 for the MPC8560. 001 The RapidIO controller supports 34-bit local addresses. 010 Reserved 011 The RapidIO controller supports 50- and 34-bit local addresses. 100 Reserved 101 The RapidIO controller supports 66- and 34-bit local addresses. 110 Reserved 111 The RapidIO controller supports 60-, 50- and 34-bit local addresses.

16.3.1.6 Switch Port Information Capability Register (SPICAR)

The SPICAR, shown in Figure 16-8, defines the RapidIO block's switching capabilities.

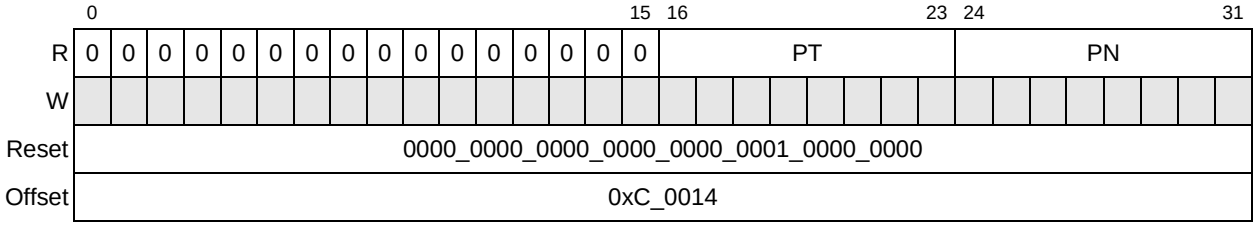


Figure 16-8. Switch Port Information Capability Register (SPICAR)

Table 16-12 describes the fields of the SPICAR.

Table 16-12. SPICAR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–23	PT	Port total. The RapidIO controller has one RapidIO port. (SPICAR[PT] = 0x01)
24–31	PN	Port number. This is the port number from which this register was read. This field is 0 since the MPC8560 has only one port.

16.3.1.7 Source Operations Capability Register (SOCAR)

The SOCAR, shown in Figure 16-9, reflects the set of RapidIO operations that this device can initiate. See the *Parallel RapidIO Intconnect Specification* for details.

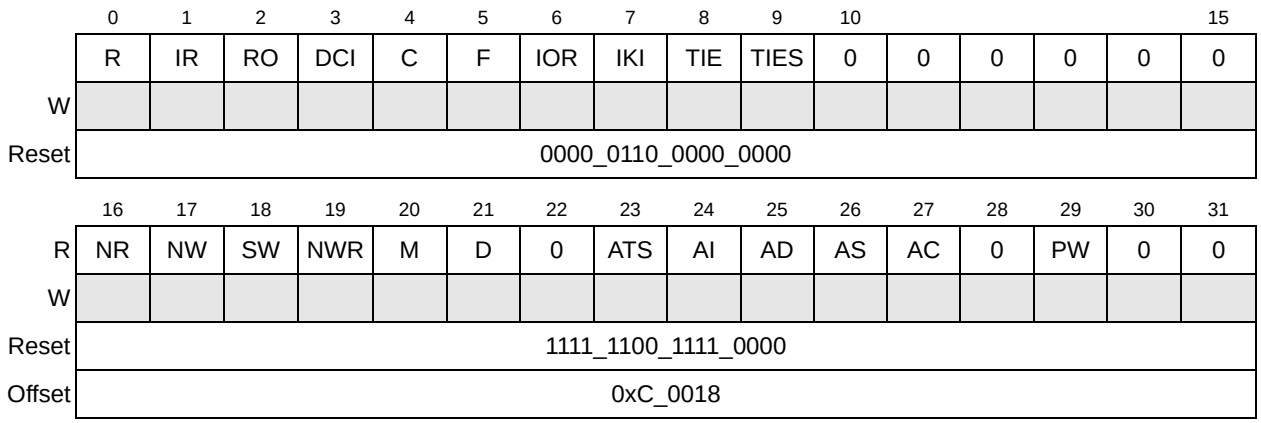


Figure 16-9. Source Operations Capability Register (SOCAR)

Table 16-13 describes the fields of the SOCAR.

Table 16-13. SOCAR Field Descriptions

Bits	Name	Description
0	R	Read operation. Cleared for the MPC8560. 0 The RapidIO controller cannot initiate a read operation. 1 The RapidIO controller can initiate a read operation.
1	IR	Iread operation. Cleared for the MPC8560. 0 The RapidIO controller cannot initiate an iread operation. 1 The RapidIO controller can initiate an iread operation.
2	RO	Read to own operation. Cleared for the MPC8560. 0 The RapidIO controller cannot initiate a read to own operation. 1 The RapidIO controller can initiate a read to own operation.
3	DCI	Dkill operation. Cleared for the MPC8560. 0 The RapidIO controller cannot initiate a dkill operation. 1 The RapidIO controller can initiate a dkill operation.
4	C	Castout operation. Cleared for the MPC8560. 0 The RapidIO controller cannot initiate a castout operation. 1 The RapidIO controller can initiate a castout operation.
5	F	Flush operation. (Return globally-shared cache line to memory.) Set for the MPC8560. 0 The RapidIO controller cannot initiate a flush operation. 1 The RapidIO controller can initiate a flush operation.
6	IOR	I/O read operation. (Read a non-cachable copy of a globally-shared cache line.) Set for the MPC8560. 0 The RapidIO controller cannot initiate an I/O read operation. 1 The RapidIO controller can initiate an I/O read operation.
7	IKI	Ikill operation. Cleared for the MPC8560. 0 The RapidIO controller cannot initiate an ikill operation. 1 The RapidIO controller can initiate an ikill operation.
8	TIE	TLBIE operation. Cleared for the MPC8560. 0 The RapidIO controller cannot initiate a TLBIE operation. 1 The RapidIO controller can initiate a TLBIE operation.
9	TIES	TLBSYNC operation. Cleared for the MPC8560. 0 The RapidIO controller cannot initiate a TLBSYNC operation. 1 The RapidIO controller can initiate a TLBSYNC operation.
10–15	—	Reserved
16	NR	Nread operation. (Read non-sharable memory.) Set for the MPC8560. 0 The RapidIO controller cannot initiate an nread operation. 1 The RapidIO controller can initiate an nread operation.
17	NW	Nwrite operation. (Write non-sharable memory.) Set for the MPC8560. 0 The RapidIO controller cannot initiate an nwrite operation. 1 The RapidIO controller can initiate an nwrite operation.
18	SW	Swrite operation. (Write non-sharable memory.) Set for the MPC8560. 0 The RapidIO controller cannot initiate an swrite operation. 1 The RapidIO controller can initiate an swrite operation.

Table 16-13. SOCAR Field Descriptions (continued)

Bits	Name	Description
19	NWR	Nwrite_r operation. (Write non-sharable memory.) Set for the MPC8560. 0 The RapidIO controller cannot initiate an nwrite_r operation. 1 The RapidIO controller can initiate an nwrite_r operation.
20	M	Message operation. (Write to port.) Set for the MPC8560. 0 The RapidIO controller cannot initiate a message operation. 1 The RapidIO controller can initiate a message operation.
21	D	Doorbell operation. (Generate and interrupt.) Set for the MPC8560. 0 The RapidIO controller cannot initiate a doorbell operation. 1 The RapidIO controller can initiate a doorbell operation.
22	—	Reserved
23	ATS	Atomic test and swap operation. Cleared for the MPC8560. 0 The RapidIO controller can initiate an atomic test and swap operation. 1 The RapidIO controller cannot initiate an atomic test and swap operation.
24	AI	Atomic_inc operation. Set for the MPC8560. 0 The RapidIO controller cannot initiate an atomic_inc operation. 1 The RapidIO controller can initiate an atomic_inc operation.
25	AD	Atomic_dec operation. Set for the MPC8560. 0 The RapidIO controller cannot initiate an atomic_dec operation. 1 The RapidIO controller can initiate an atomic_dec operation.
26	AS	Atomic_set operation. Set for the MPC8560. 0 The RapidIO controller cannot initiate an atomic_set operation. 1 The RapidIO controller can initiate an atomic_set operation.
27	AC	Atomic_clr operation. Set for the MPC8560. 0 The RapidIO controller cannot initiate an atomic_clr operation. 1 The RapidIO controller can initiate an atomic_clr operation.
28	—	Reserved
29	PW	Port-write operation. Cleared for the MPC8560. 0 The RapidIO controller cannot initiate a port-write operation. 1 The RapidIO controller can initiate a port-write operation.
30–31	—	Reserved

16.3.1.8 Destination Operations Capability Register (DOCAR)

The DOCAR, shown in Figure 16-10, describes RapidIO I/O operations that this device can service. See the *Parallel RapidIO Intconnect Specification* for additional details.

	0	1	2	3	4	5	6	7	8	9	10					15
	R	IR	RO	DCI	C	F	IOR	IKI	TIE	TIES	0	0	0	0	0	0
W																
Reset	0000_0110_0000_0000															

	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	1111_1100_1111_0100															
Offset	0xC_001C															

Figure 16-10. Destination Operations Capability Register (DOCAR)

Table 16-14 describes the fields of the DOCAR.

Table 16-14. DOCAR Field Descriptions

Bits	Name	Description
0	R	Read operation. Cleared for the MPC8560. 0 The RapidIO controller cannot service a read operation. 1 The RapidIO controller can service a read operation.
1	IR	Iread operation. Cleared for the MPC8560. 0 The RapidIO controller cannot service an iread operation. 1 The RapidIO controller can service an iread operation.
2	RO	Read to own operation. Cleared for the MPC8560. 0 The RapidIO controller cannot service a read to own operation. 1 The RapidIO controller can service a read to own operation.
3	DCI	Dkill operation. Cleared for the MPC8560. 0 The RapidIO controller cannot service a dkill operation. 1 The RapidIO controller can service a dkill operation.
4	C	Castout operation. Cleared for the MPC8560. 0 The RapidIO controller cannot service a castout operation. 1 The RapidIO controller can service a castout operation.
5	F	Flush operation. (Return globally-shared cache line to memory.) Set for the MPC8560. 0 The RapidIO controller cannot service a flush operation. 1 The RapidIO controller can service a flush operation.
6	IOR	I/O read operation. (Read a non-cachable copy of a globally-shared cache line.) Set for the MPC8560. 0 The RapidIO controller cannot service an I/O read operation. 1 The RapidIO controller can service an I/O read operation.
7	IKI	Ikill operation. Cleared for the MPC8560. 0 The RapidIO controller cannot service an ikill operation. 1 The RapidIO controller can service an ikill operation.
8	TIE	TLBIE operation. Cleared for the MPC8560. 0 The RapidIO controller cannot service a TLBIE operation. 1 The RapidIO controller can service a TLBIE operation.

Table 16-14. DOCAR Field Descriptions (continued)

Bits	Name	Description
9	TIES	TLBSYNC operation. Cleared for the MPC8560. 0 The RapidIO controller cannot service a TLBSYNC operation. 1 The RapidIO controller can service a TLBSYNC operation.
10–15	—	Reserved
16	NR	Nread operation. (Read non-sharable memory.) Set for the MPC8560. 0 The RapidIO controller cannot service an nread operation. 1 The RapidIO controller can service an nread operation.
17	NW	Nwrite operation. (Write non-sharable memory.) Set for the MPC8560. 0 The RapidIO controller cannot service an nwrite operation. 1 The RapidIO controller can service an nwrite operation.
18	SW	Swrite operation. (Write non-sharable memory.) Set for the MPC8560. 0 The RapidIO controller cannot service an swrite operation. 1 The RapidIO controller can service an swrite operation.
19	NWR	Nwrite_r operation. (Write non-sharable memory.) Set for the MPC8560. 0 The RapidIO controller cannot service an nwrite_r operation. 1 The RapidIO controller can service an nwrite_r operation.
20	M	Message operation. (Write to port.) Set for the MPC8560. 0 The RapidIO controller cannot service a message operation. 1 The RapidIO controller can service a message operation.
21	D	Doorbell operation. (Generate and interrupt.) Set for the MPC8560. 0 The RapidIO controller cannot service a doorbell operation. 1 The RapidIO controller can service a doorbell operation.
22	—	Reserved
23	ATS	Atomic test and swap operation. Cleared for the MPC8560. 0 The RapidIO controller can service an atomic test and swap operation. 1 The RapidIO controller cannot service an atomic test and swap operation.
24	AI	Atomic_inc operation. Set for the MPC8560. 0 The RapidIO controller cannot service an atomic_inc operation. 1 The RapidIO controller can service an atomic_inc operation.
25	AD	Atomic_dec operation. Set for the MPC8560. 0 The RapidIO controller cannot service an atomic_dec operation. 1 The RapidIO controller can service an atomic_dec operation.
26	AS	Atomic_set operation. Set for the MPC8560. 0 The RapidIO controller cannot service an atomic_set operation. 1 The RapidIO controller can service an atomic_set operation.
27	AC	Atomic_clr operation. Set for the MPC8560. 0 The RapidIO controller cannot service an atomic_clr operation. 1 The RapidIO controller can service an atomic_clr operation.
28	—	Reserved
29	PW	Port-write operation. Set for the MPC8560. 0 The RapidIO controller cannot service a port-write operation. 1 The RapidIO controller can service a port-write operation.
30–31	—	Reserved

16.3.1.9 Mailbox Command and Status Register (MSR)

The MSR, shown in Figure 16-11, reflects the status of the RapidIO mailbox controller on this device. Note that the RapidIO controller only supports Mailbox 0.

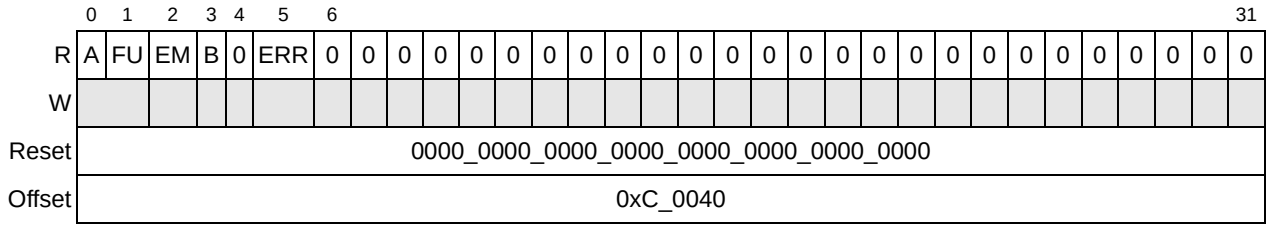


Figure 16-11. Mailbox Status Register (MSR)

Table 16-15 describes the fields of the MSR.

Table 16-15. MSR Field Definitions

Bits	Name	Description
0	A	Available 0 Mailbox 0 is not ready to accept messages. All incoming message transactions return error responses. 1 Mailbox 0 is initialized and ready to accept messages.
1	FU	Full 0 Mailbox 0 is not full. 1 Mailbox 0 is full. New messages are retried.
2	EM	Empty 0 Mailbox 0 contains outstanding messages. 1 Mailbox 0 contains no outstanding messages.
3	B	Busy 0 Mailbox 0 is not busy processing a message. 1 Mailbox 0 is busy processing a message. New message operations return retry responses.
4	—	Reserved
5	ERR	Error 0 Mailbox 0 has not received an illegal message operation. 1 Mailbox 0 received an illegal message operation. All incoming message transactions return error responses.
6–31	—	Reserved

16.3.1.10 Port-Write and Doorbell Command and Status Register (PWDCSR)

The PWDCSR, shown in Figure 16-12, reflects the status of the RapidIO doorbell and port-write hardware on this device. Additional details can be found in the *Parallel RapidIO Intconnect Specification* in the sections titled “Doorbell CSR” and “Write Port CSR.”

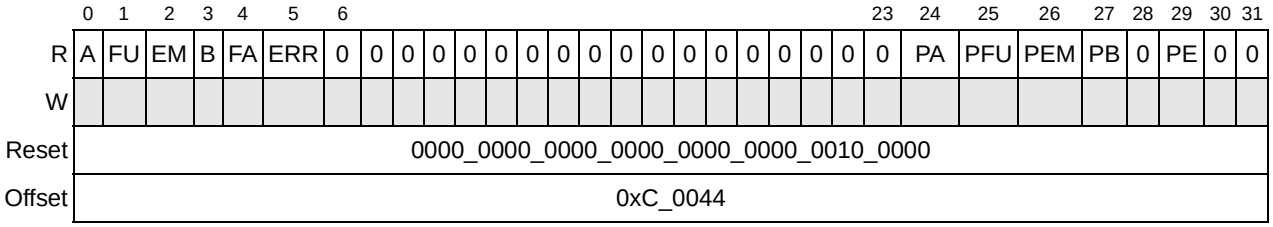


Figure 16-12. Port-Write and Doorbell Command and Status Register (PWDCSR)

Table 16-16 describes the fields of the PWDCSR.

Table 16-16. PWDCSR Field Descriptions

Bits	Name	Description
0	A	Available 0 Doorbell unit is not ready to accept doorbell messages. All incoming doorbell transactions return error responses. 1 Doorbell unit is ready to accept doorbell messages
1	FU	Full 0 Doorbell unit is not full 1 Doorbell unit is full. New doorbell messages are retried.
2	EM	Empty 0 Doorbell unit has outstanding doorbell messages 1 Doorbell unit has no outstanding doorbell messages
3	B	Busy 0 Doorbell unit is not busy processing a doorbell message 1 Doorbell unit is busy processing a doorbell message. Incoming transactions are retried.
4	FA	Failure. (Defined by the RapidIO specification but not implemented in the *MPC8560*.) 0 Doorbell unit has not encountered an internal error 1 Doorbell unit has encountered an internal error
5	ERR	Error 0 Doorbell unit has not encountered an error condition 1 Doorbell unit encountered an error condition. Incoming doorbell transactions return error responses.
6–23	—	Reserved
24	PA	Port-write unit available 0 Port-write unit is not available. All incoming port-write packets are discarded. 1 Port-write unit is initialized and ready to accept a port-write packet
25	PFU	Port-write unit full 0 Port-write unit is not full 1 Port-write unit is full. All incoming port-write packets are discarded.
26	PEM	Port-write unit empty 0 Port-write unit contains outstanding port-write data 1 Port-write unit contains no outstanding port-write data
27	PB	Port-write unit busy 0 Port-write unit is not busy 1 Port-write unit is busy processing a port-write packet. Incoming port-write packets are discarded.
28	—	Reserved

Table 16-16. PWDCSR Field Descriptions (continued)

Bits	Name	Description
29	PE	Port-write unit error 0 Received port-write packet contains no errors, for example, data is of legal size (64 bytes or less) 1 Received port-write packet contains fatal errors, for example, data is of illegal size (greater than 64 bytes). All incoming port-write transactions are discarded.
30–31	—	Reserved

16.3.1.11 Processing Element Logic Layer Control Command and Status Register (PELLCCSR)

The PELLCCSR, shown in Figure 16-13, is a read-only register. Additional details of this register can be found in the *Parallel RapidIO Intconnect Specification*.

	0																											28	29	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EAC	
W																														
Reset	0000_0000_0000_0000_0000_0000_0000_0001																													
Offset	0xC_004C																													

Figure 16-13. Processing Element Logic Layer Control Command and Status Register (PELLCCSR)

Table 16-17 describes the fields of the PELLCCSR.

Table 16-17. PELLCCSR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29–31	EAC	Extended addressing control. The RapidIO controller supports 34-bit addresses (read-only).

16.3.1.12 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 14 msbs of the double-word address, which represents the base address for the device's register space.

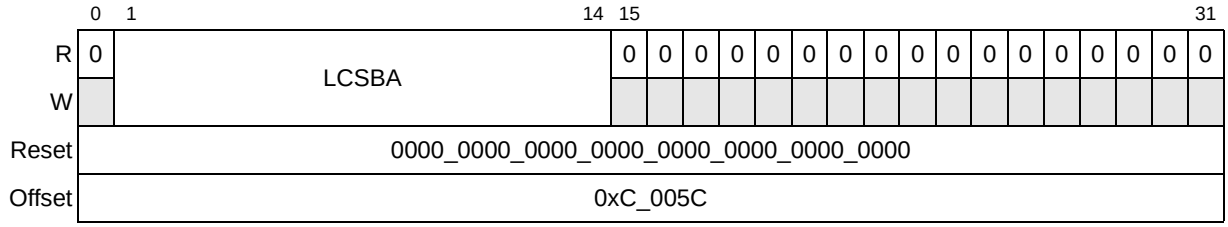


Figure 16-14. Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)

Table 16-18 describes the fields of the LCSBA1CSR.

Table 16-18. LCSBA1CSR Field Descriptions

Bits	Name	Description
0	—	Reserved
1–14	LCSBA	Local configuration space base address. These bits correspond to the 14 most-significant bits of the 34-bit RapidIO address space.
15–31	—	Reserved

16.3.1.13 Base Device ID Command and Status Register (BDIDCSR)

Figure 16-15 shows the BDIDSCR.

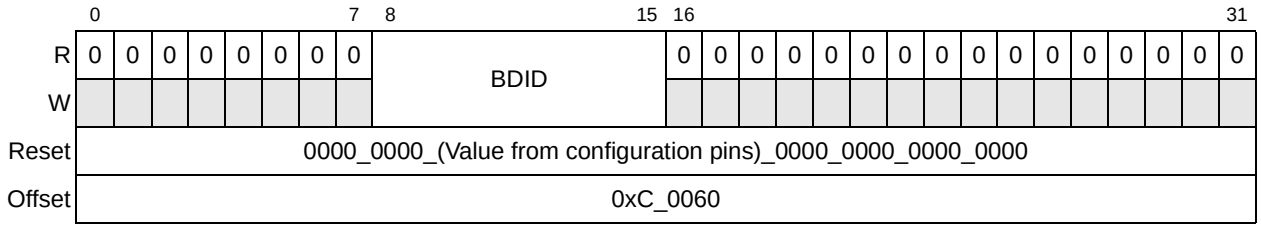


Figure 16-15. Base Device ID Command and Status Register (BDIDCSR)

Table 16-19 describes the fields of the BDIDCSR.

Table 16-19. BDIDCSR Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	BDID	Base device ID (RapidIO device ID). Value read from reset configuration pins. For more information on the RapidIO device ID, see Section 4.4.3.11, “RapidIO Device ID.”
16–31	—	Reserved

16.3.1.14 Host Base Device ID Lock Command and Status Register (HBDIDLCSR)

The host base device ID lock CSR, shown in Figure 16-16, 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 that provides a lock function. After HBDID is written, subsequent writes to the field are ignored, except when the written value matches the value contained in the field. In this case, the register is reinitialized to 0xFFFF. After HBDID is written, 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.

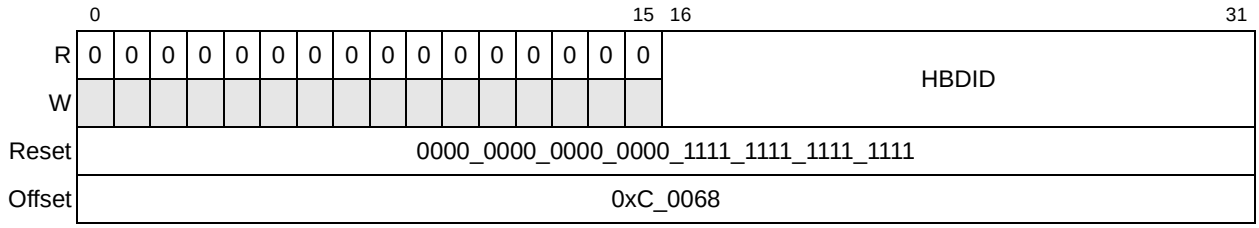


Figure 16-16. Host Base Device ID Lock Command and Status Register (HBDIDLCSR)

Table 16-20 describes the fields of the HBDIDLCSR.

Table 16-20. HBDIDLCSR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	HBDID	Host base device ID. This is the base device ID for the processing element that is initializing this processing element. When unlocked, this field is write-able and locks immediately after being written. When locked, this field is only write-able when written with the value it already contains, at which point it becomes unlocked.

16.3.1.15 Component Tag Command and Status Register (CTCSR)

The CTCSR contains a component tag value for the RapidIO block and can be assigned by software when the device is initialized. Figure 16-17 shows the CTCSR.

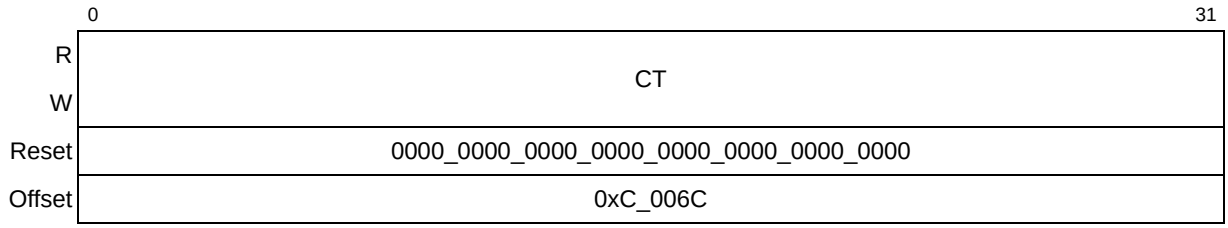


Figure 16-17. Component Tag Command and Status Register (CTCSR)

Table 16-21 describes the field of the CTCSR.

Table 16-21. CTCSR Field Descriptions

Bits	Name	Description
0–31	CT	Component tag. Identifying component tag for the RapidIO block.

16.3.1.16 8/16 LP-LVDS Port Maintenance Block Header 0 Command and Status Register (PMBH0CSR)

The port maintenance block header 0 command and status register, shown in Figure 16-18, contains the extended features pointer to the next extended features block and the extended features ID that identifies this as the generic end point port maintenance block header.

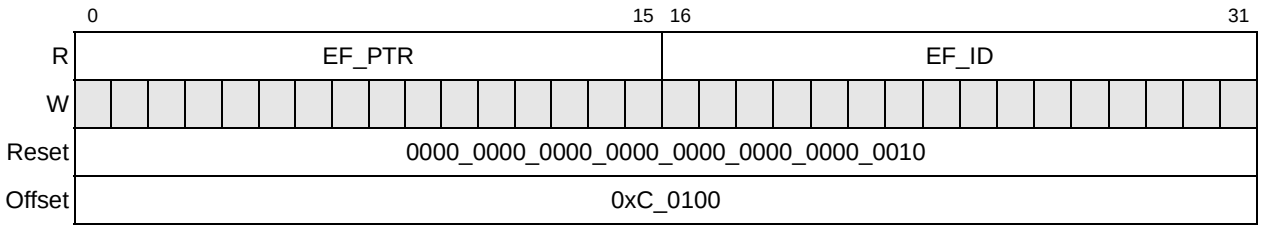


Figure 16-18. 8/16 LP-LVDS Port Maintenance Block Header 0 Command and Status Register (PMBH0CSR)

Table 16-22 describes the fields of the PMBH0CSR.

Table 16-22. PMBH0CSR Field Descriptions

Bits	Name	Description
0–15	EF_PTR	Extended features pointer. Points to the next extended features block. 0x0000 for the MPC8560.
16–31	EF_ID	Extended features ID. Identifies this as the generic end point port maintenance block header. 0x0002 for the MPC8560.

16.3.1.17 Port Link Time-Out Control Command and Status Register (PLTOCCSR)

The PLTOCCSR, shown in Figure 16-19, contains the time-out timer value used to monitor link events, for example, between sending a packet and receiving the corresponding acknowledge. Alternately, it can contain the value of the timer time-out between sending a link-request and receiving the corresponding link-response. The reset value is the maximum time-out interval and represents between three and five seconds.

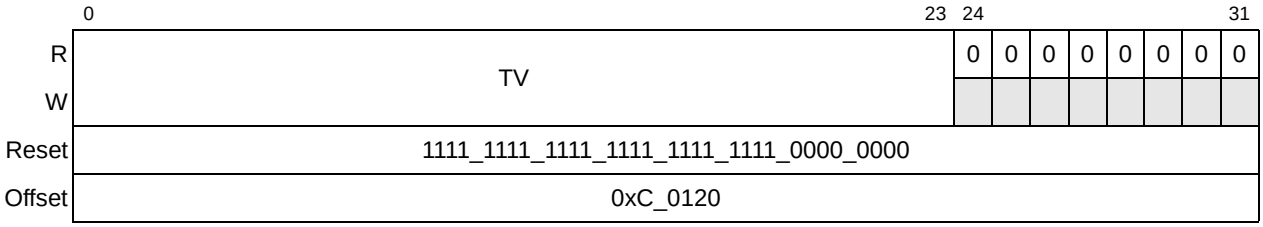


Figure 16-19. Port Link Time-Out Control Command and Status Register (PLTOCCSR)

Table 16-23 describes the fields of the PLTOCCSR.

Table 16-23. PLTOCCSR Field Descriptions

Bits	Name	Description
0–23	TV	Time-out value. This value is expressed in platform clock cycles.
24–31	—	Reserved

16.3.1.18 Port Response Time-Out Control Command and Status Register (PRTOCCSR)

PRTOCCSR, shown in Figure 16-20, contains the time-out timer value used to monitor port events, for instance, between sending a request packet and receiving the corresponding response packet.

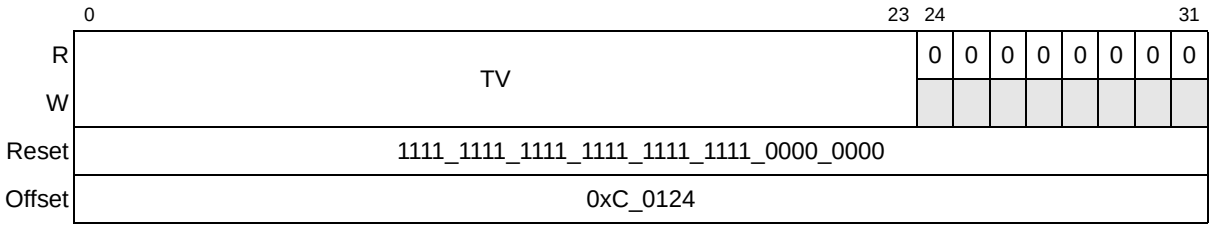


Figure 16-20. Port Response Time-Out Control Command and Status Register (PRTOCCSR)

Table 16-24 describes the fields of the PRTOCCSR.

Table 16-24. PRTOCCSR Field Descriptions

Bits	Name	Description
0–23	TV	Time-out value. This value is expressed in platform clock cycles.
24–31	—	Reserved

16.3.1.19 Port General Control Command and Status Register (PGCCSR)

The PGCCSR, shown in Figure 16-21, reflects the state of the RapidIO port.

	0	1	2	3																											
																															31
R	H	M	D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																															
Reset	(Value from configuration pins)_0000_0000_0000_0000_0000_0000_0000																														
Offset	0xC_013C																														

Figure 16-21. Port General Control Command and Status Register (PGCCSR)

Table 16-25 describes the fields of the PGCCSR.

Table 16-25. PGCCSR Field Descriptions

Bits	Name	Description
0	H	A host device is a device that is responsible for system exploration, initialization, and maintenance. Agent (or slave) devices are initialized by host devices. Initialized from the host/agent configuration pin; If host is sampled, H is set as host. 0 Agent device 1 Host device
1	M	Master enable bit. Controls whether or not a device is allowed to issue requests into the system. If the master enable is not set, the device may only respond to requests. Initialized from the host/agent configuration pin. Out of reset, if configured as a host (PGCCSR[H] is set), M and D are also set. If configured as an agent (PGCCSR[H] is cleared), M and D are initialized by the host of the system. 0 Device is not enabled to issue requests into the system 1 Device is enabled to issue requests into the system
2	D	Discovery enable bit. This device has been located by the processing element responsible for system configuration. Initialized from the host/agent configuration pin. Out of reset, if configured as a host (PGCCSR[H] is set), bits M and D are also set. If configured as an agent (PGCCSR[H] is cleared), bits M and D are initialized by the host of the system. 0 Device has not been discovered by system host 1 Device has been discovered by system host
3–31	—	Reserved

16.3.1.20 Port Link Maintenance Request Command and Status Register (PLMREQCSR)

A write to the PLMREQCSR, shown in Figure 16-22, generates a link-request control symbol on the RapidIO port interface. See the *Parallel RapidIO Intconnect Specification* for descriptions of the reset command, input-status command and send-training commands.

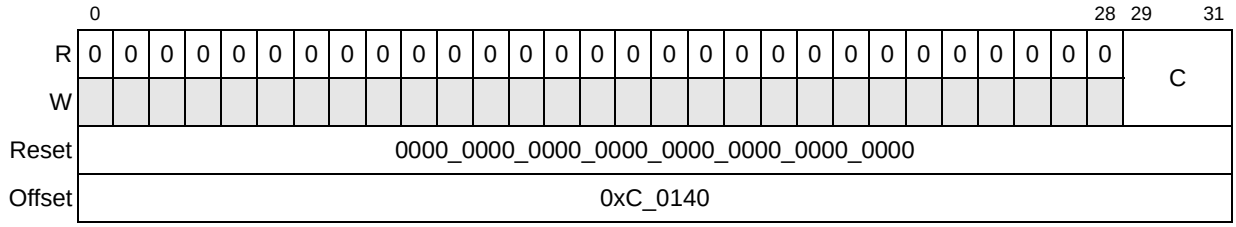


Figure 16-22. Port Link Maintenance Request Command and Status Register (PLMREQCSR)

Table 16-26 describes the fields of the PLMREQCSR.

Table 16-26. PLMREQCSR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29–31	C	Command. This is the command to be sent in the link-request control symbol. See Table 16-85 for command symbol formats.

16.3.1.21 Port Link Maintenance Response Command and Status Register (PLMRESPCSR)

A read from the PLMRESPCSR, shown in Figure 16-23, returns the status received in a link-response control symbol.

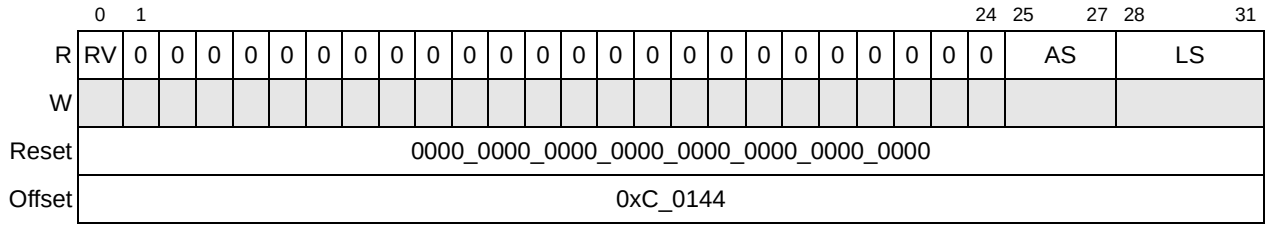


Figure 16-23. Port Link Maintenance Response Command and Status Register (PLMRESPCSR)

Table 16-27 describes the fields of the PLMRESPCSR.

Table 16-27. PLMRESPCSR Field Descriptions

Bits	Name	Description
0	RV	Response valid (clear on read) 0 If the link-request caused a link-response, the link-response has not yet been received. If the link-request did not cause a link-response, the link-request has not yet been transmitted. 1 If the link-request caused a link-response, the link-response has been received and the status fields are valid. If the link-request did not cause a link-response, the link-request has been transmitted.
1–24	—	Reserved

Table 16-27. PLMRESPCSR Field Descriptions (continued)

Bits	Name	Description
25–27	AS	AckID status. This is the ackID status field from the link-response control symbol.
28–31	LS	Link status. This is the link status field from the link-response control symbol.

16.3.1.22 Port Local AckID Status Command and Status Register (PLASCSR)

A read from the PLASCSR returns the local ackID status for both the outbound and inbound RapidIO ports. Figure 16-24 shows the PLASCSR.

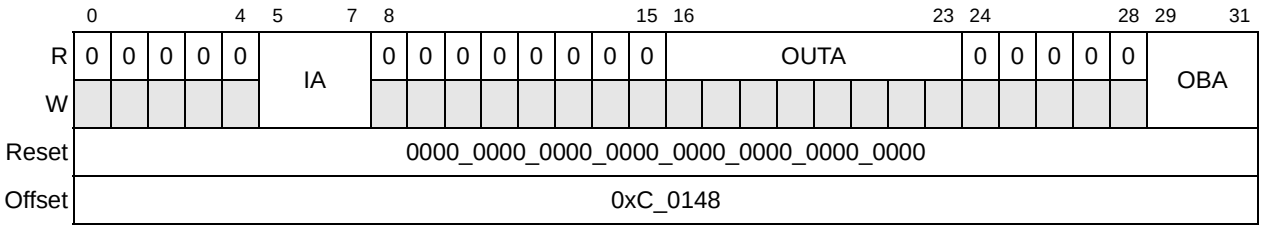


Figure 16-24. Port Local AckID Status Command and Status Register (PLASCSR)

Table 16-28 describes the fields of the PLASCSR.

Table 16-28. 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 in 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 ti manually implement error recovery. OBA can only be written by software when the RapidIO port is enabled.

16.3.1.23 Port Error and Status Command and Status Register (PESCSR)

The PESCSR contains port error and status information. Figure 16-25 shows the PESCSR.

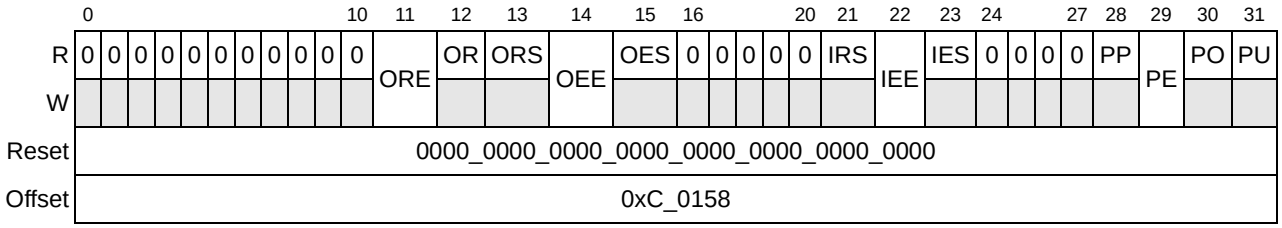


Figure 16-25. Port Error and Status Command and Status Register (PESCSR)

Table 16-29 describes the fields of the PESCSR.

Table 16-29. PESCSR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	ORE	Output port retry condition 0 Output port operating normally 1 Output port encountered a retry condition. Set when ORS is set. (Bit reset, write-one-to-clear)
12	OR	Output port retry (read-only field). (Bit reset, write-one-to-clear) 0 Output port received a packet-accepted or packet-not-accepted control symbol 1 Output port received a packet retry and cannot make forward progress. Set when ORS is set
13	ORS	Output port retry stop (read-only field) 0 Output port operating normally 1 Output port is stopped due to a retry
14	OEE	Output port error encounter. (Bit reset, write-one-to-clear) 0 Output port operating normally 1 Output port encountered an error. Set when OES is set
15	OES	Output port error stop (read-only field) 0 Output port operating normally 1 Output port is stopped due to a transmission error
16–20	—	Reserved
21	IRS	Input port retry stop (read-only field) 0 Input port operating normally 1 Input port is stopped due to a retry condition
22	IEE	Input port error encounter. (Bit reset, write-one-to-clear) 0 Input port operating normally 1 Input port encountered a transmission error. Set when IES is set
23	IES	Input port error stop (read-only field) 0 Input port operating normally 1 Input port is stopped due to a transmission error
24–27	—	Reserved

Table 16-29. PESCSR Field Descriptions (continued)

Bits	Name	Description
28	PP	Input clock toggling (read-only field) Note: Set shortly after reset, due to asynchronous interface crossing and clock detection delay 0 Input clock pin is not toggling 1 Input clock pin is toggling. (There is a device attached.)
29	PE	Input port error. (Bit reset, write-one-to-clear) 0 Input port operating properly 1 Input port encountered an unrecoverable error or output port failed error recover
30	PO	Port initialization (read-only) 0 Input or output port not initialized 1 Input and output ports are initialized and can communicate with the adjacent device.
31	PU	Port training mode (read-only) Note: Set shortly after reset due to asynchronous interface crossing delay 0 Input and output ports initialized 1 Input and output ports are not initialized and are in training mode

16.3.1.24 Port Control Command and Status Register (PCCSR)

The PCCSR contains bits used for port control. Figure 16-26 shows the PCCSR.

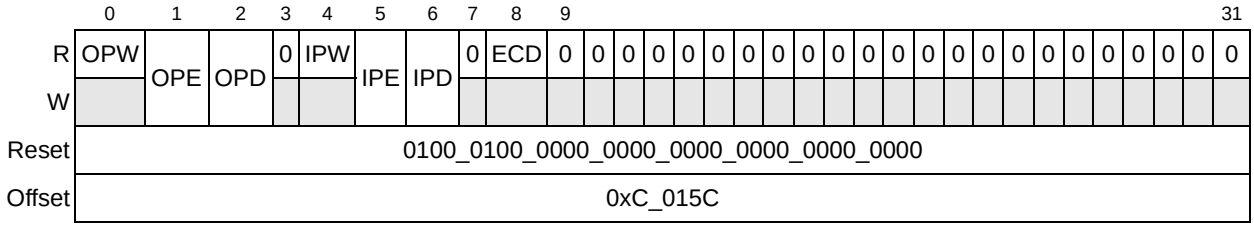


Figure 16-26. Port Control Command and Status Register (PCCSR)

Table 16-30 describes the fields of the PCCSR.

Table 16-30. 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. Must equal the value of PCCSR[IPE] for the RapidIO controller to function properly. 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
2	OPD	Output port drivers disable 0 Input receiver and/or output driver are enabled 1 Both the input receivers and output drivers are disabled.
3	—	Reserved

Table 16-30. PCCSR Field Descriptions (continued)

Bits	Name	Description
4	IPW	Operating width of the input port (read-only field) 0 Port operating in 8-bit mode 1 Reserved
5	IPE	Input port receive enable. Must equal the value of PCCSR[OPE] for the RapidIO controller to function properly 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
6	IPD	Input port receivers disable 0 Input receiver and/or output driver are enabled 1 Both the input receivers and output drivers are disabled.
7	—	Reserved
8	ECD	Error checking disable. 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.
9–31	—	Reserved

16.3.2 Implementation Registers

Following are the RapidIO implementation registers. These are registers specific to this particular RapidIO implementation and, therefore, do not appear in the *Parallel RapidIO Intconnect Specification*.

16.3.2.1 General Registers

16.3.2.1.1 Configuration Register (CR)

The CR contains configuration information regarding the behavior of the RapidIO controller. Figure 16-27 shows the CR.

	0																												30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AA
W																														
Reset	0000_0000_0000_0000_0000_0000_0000_0001																													
Offset	0xD_0000																													

Figure 16-27. Configuration Register (CR)

Table 16-31 describes the fields of the CR.

Table 16-31. CR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	AA	Accept all. Note: Must initially be cleared by the user as part of initialization 0 Detect an error upon receiving a packet with a target ID not equal to its own 1 All packets are accepted without checking the target ID.

16.3.2.1.2 Port Configuration Register (PCR)

The PCR contains configuration information regarding port behavior of the RapidIO controller. Figure 16-28 shows the PCR.

	0																							26	27	28			31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CCE	0	0	0	0
W																													
Reset	0000_0000_0000_0000_0000_0000_0001_0000																												
Offset	0xD_0010																												

Figure 16-28. Port Configuration Register (PCR)

Table 16-32 describes the fields of the PCR.

Table 16-32. PCR Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	CCE	CRC checking enable. Note that a read that attempts to access an unmapped target causes the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, PCR[CCE] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).” 0 No CRC is checked on received packets. 1 CRC is checked on received packets.
28–31	—	Reserved

16.3.2.1.3 Port Error Injection Register (PEIR)

RapidIO error injection registers, shown in Figure 16-29, offer a simple way to inject errors into an outbound packet or a control symbol. The 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 cleared if error injection is enabled and SE is not set. The error is injected continuously if both EN and SE are set. It stops if software clears the enable bit. After setting the enable bit, software can monitor the injection of the error by polling the register and observing the enable bit being cleared.

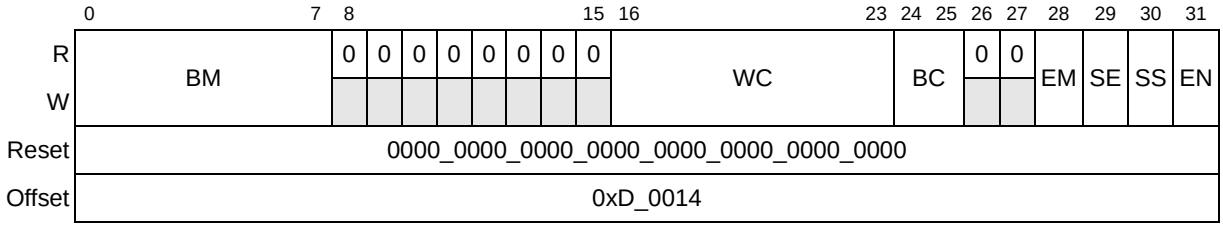


Figure 16-29. Port Error Injection Register (PEIR)

Table 16-33 describes the fields of the PEIR.

Table 16-33. PEIR Field Descriptions

Bits	Name	Description
0–7	BM	Byte mask. Indicates on which bit of the byte to inject error
8–15	—	Reserved
16–23	WC	Word count. Number of words to count before injecting error
24–25	BC	Byte count. Byte within the specified word count on which to inject error
26–27	—	Reserved
28	EM	Error mirror. Setting EM when BC is 00 causes the error to be mirrored to 10 and when BC is 01, the error is mirrored to 11. This gives better flexibility for error injection. For example, an EOP symbol can be converted to a STOMP symbol using this bit. Setting EM when BC is 10 or 11 causes an illegal condition.
29	SE	Software enable. If SE and EN are set, an error continues to be injected until EN is cleared by software.
30	SS	Symbol select. 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.

16.3.2.2 ATMU Registers

Although the *Parallel RapidIO Intconnect 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 4 Kbytes and the largest window size is 4 Gbytes. See Section 16.7, “ATMU (Address Translation and Mapping Unit),” for more information.

16.3.2.2.1 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. This register takes on one of two formats, determined by the RDTYP and WRTYP fields in the corresponding attributes register (ROWAR*n*), one for standard operations and one for maintenance operations. See Section 16.7.1, “Outbound ATMU Translation,” for more information. Figure 16-30 shows the ROWTARs as used in standard operations.

	0	1	2–9	10–11	12–31
R	0	0	TRGTID	TREXAD	TRAD
W					
Reset	0000_0000_0000_0000_0000_0000_0000_0000				
Offset	0xD_0C00, 0xD_0C20, 0xD_0C40, 0xD_0C60, 0xD_0C80, 0xD_0CA0, 0xD_0CC0, 0xD_0CE0, 0xD_0D00				

Figure 16-30. RapidIO Outbound Window Translation Address Registers 0–8 for Standard Transactions (ROWTAR*n*)

Table 16-34 describes the fields of the ROWTARs for standard operations.

Table 16-34. ROWTAR*n* Standard Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–9	TRGTID	Target ID for RapidIO packet
10–11	TREXAD	Translation extended address of outbound window The translation extended address correspond to bits 0–1 of the 34-bit RapidIO address.
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.

Figure 16-31 shows the ROWTARs for maintenance transactions.

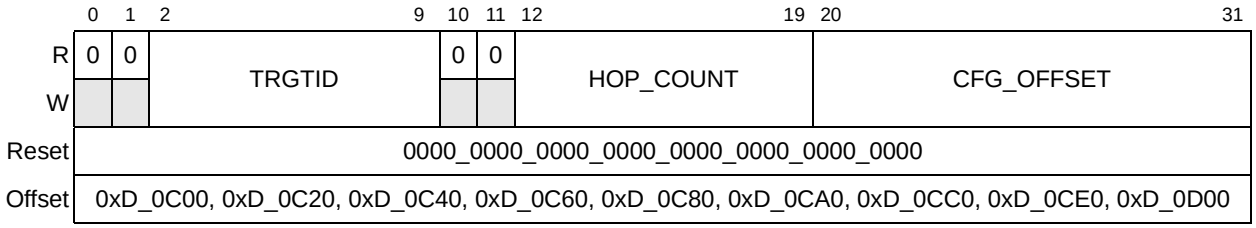


Figure 16-31. RapidIO Outbound Window Translation Address Registers 0–8 for Maintenance Transactions (ROWTAR*n*)

Table 16-35 describes ROWTARs fields for maintenance transactions. For maintenance transactions, the hop count is formed from ROWTAR[12–19]. Similarly, the upper 12 bits of the maintenance offset is formed from ROWTAR[20–31]. The rest of the maintenance offset is formed from the untranslated address.

Table 16-35. ROWTAR*n* Maintenance Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–9	TRGTID	Target ID for RapidIO Packet
10–11	—	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.

16.3.2.2.2 RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR*n*)

The RapidIO outbound window translation address registers (ROWBAR*n*), shown in Figure 16-32, select the base address for the windows that 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. Since window 0 is the default window, and default translation does not require a base address register, ROWBAR0 is not implemented.

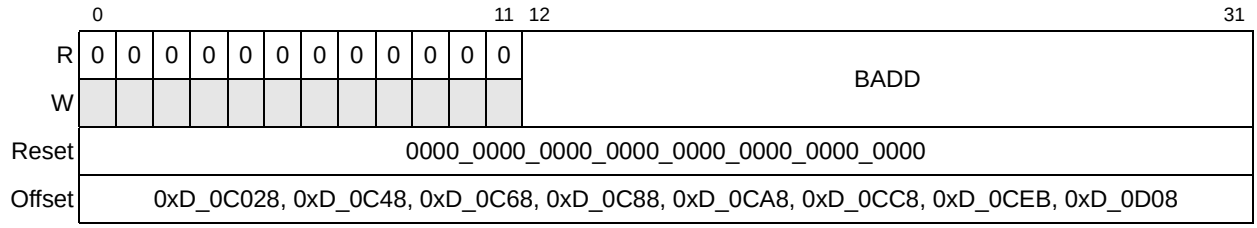


Figure 16-32. RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR*n*)

Table 16-36 describes the fields of the ROWBARs.

Table 16-36. 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.

16.3.2.2.3 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. The largest window size allowed is 4 Gbytes. Figure 16-33 and Figure 16-34 show the ROWARs.

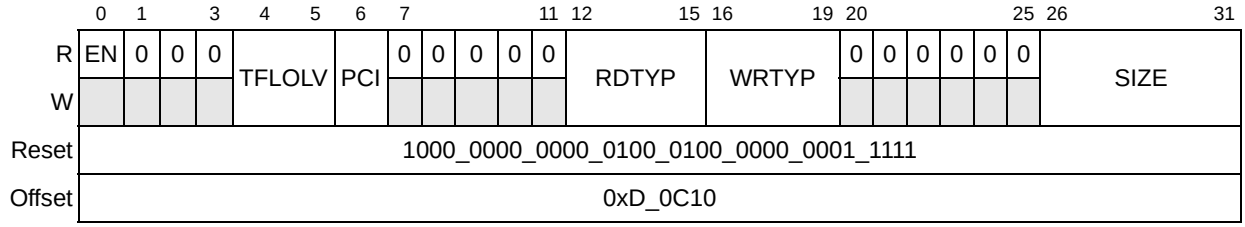


Figure 16-33. RapidIO Outbound Window Attributes Register 0 (ROWAR0)

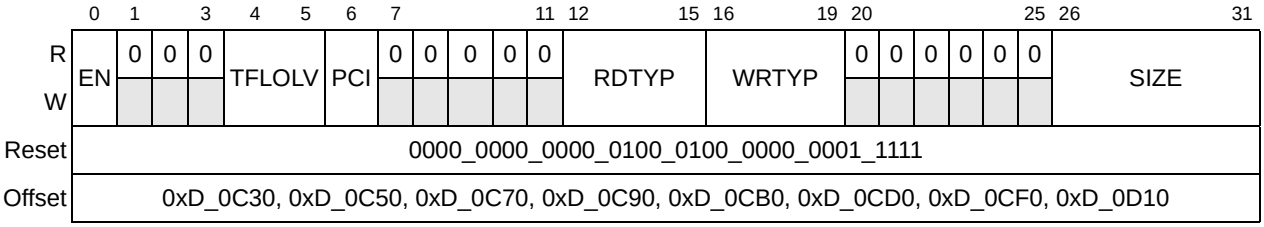


Figure 16-34. RapidIO Outbound Window Attributes Registers 1–8 (ROWAR*n*)

Table 16-37 describes the fields of the ROWARs.

Table 16-37. ROWAR*n* Field Descriptions

Bits	Name	Description
0	EN	Window address translation enable. Note that for ROWAR0 this bit is read-only and 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	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. See Section 16.7.1.2, “Outbound Special Transactions and Requirements,” for more information on maintenance transactions. 0000–0001 Reserved 0010 I/O read home 0011 Reserved 0100 nread 0101–0110 Reserved 0111 Maintenance read 1000–1011 Reserved 1100 atomic_inc 1101 atomic_dec 1110 atomic_set 1111 atomic_clr
16–19	WRTYP	Write transaction type. Transaction type to run on RapidIO interface if access is a write. See Section 16.7.1.2, “Outbound Special Transactions and Requirements,” for more information on maintenance and doorbell transactions. 0000 Reserved 0001 Flush with data 0010 Doorbell 0011 swrite 0100 nwrite 0101 nwrite_r 0110 Reserved 0111 Maintenance write 1000–1111 Reserved
20–25	—	Reserved
26–31	SIZE	Outbound window size. Outbound window size n which is the encoded 2^{n+1}-byte window size. The smallest window size is 4 Kbytes; the largest is 4 Gbytes. 00_0000–00_1010 Reserved 00_1011 4 Kbytes 00_1100 8 Kbytes ... 01_1111 4 Gbytes 10_0000–11_1111 Reserved

16.3.2.2.4 RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR*n*)

The RapidIO inbound window translation address registers (RIWTARs), shown in Figure 16-35, 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. See Section 16.7.2, “Inbound ATMU Translation,” for more information.

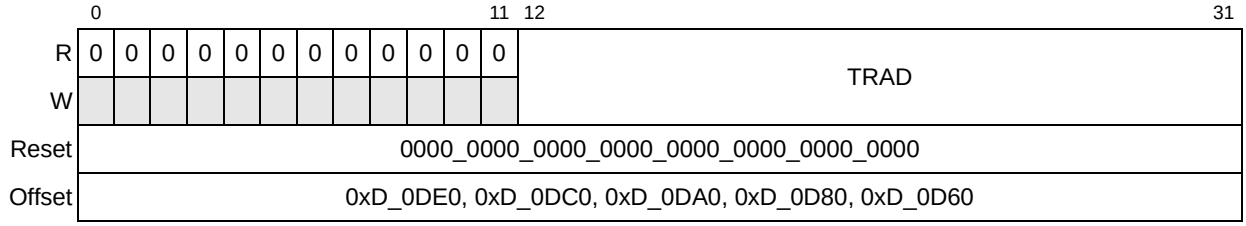


Figure 16-35. RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR*n*)

Table 16-38 describes the fields of the RIWTARs.

Table 16-38. 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.

16.3.2.2.5 RapidIO Inbound Window Base Address Registers 1–4 (RIWBAR*n*)

RapidIO inbound window translation address registers (RIWBARs), shown in Figure 16-36, select the base address for windows that are translated to an alternate system address space. Addresses for inbound transactions are compared to these windows. If such a transaction does not fall in the LCSBA1CSR window or one of these spaces, the transaction is forwarded through the window 0 register set. Since window 0 is the default and default translation does not require a base address register, RIWBAR0 is not implemented.

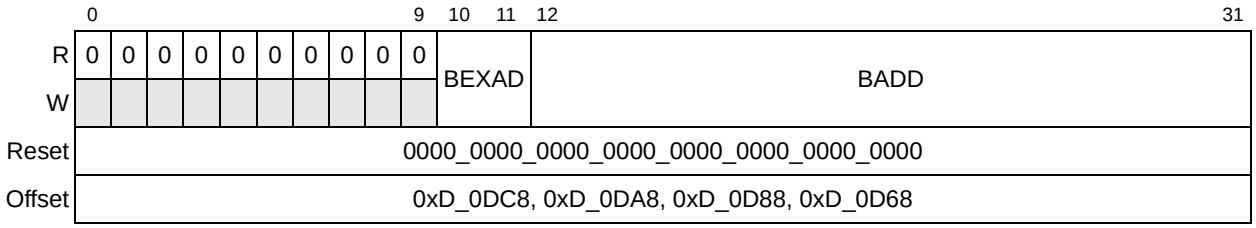


Figure 16-36. RapidIO Inbound Window Base Address Registers 1–4 (RIWBAR*n*)

Table 16-39 describes the fields of the RIWBARs.

Table 16-39. 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.

16.3.2.2.6 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. The largest window size allowed is 4 Gbytes. Figure 16-38 and Figure 16-37 show the RIWARs.

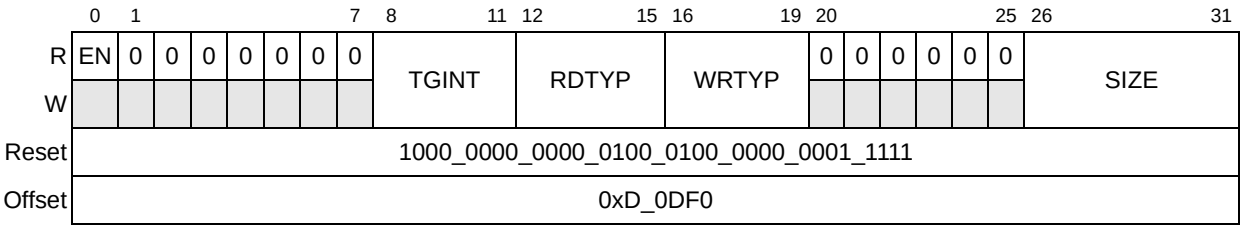


Figure 16-37. RapidIO Inbound Window Attributes Register 0 (RIWAR0)

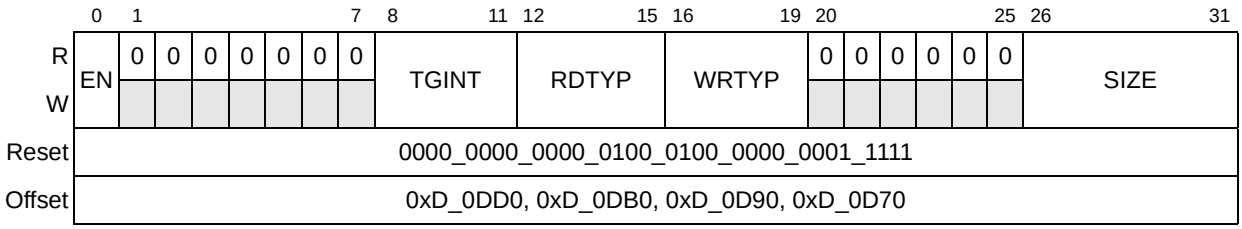


Figure 16-38. RapidIO Inbound Window Attributes Registers 1–4 (RIWAR*n*)

Table 16-40 describes the fields of the RIWARs.

Table 16-40. RIWAR*n* Field Descriptions

Bits	Name	Description
0	EN	Window address translation enable. Note that for RIWAR0 the enable bit is read only and hardwired to 1. 0 Address translation disabled 1 Address translation enabled
1–7	—	Reserved

Table 16-40. RIWAR*n* Field Descriptions (continued)

Bits	Name	Description
8–11	TGINT	Target interface 0000 PCI/PCI-X 0001–1110 Reserved 1111 Local memory (DDR SDRAM, local bus controller (LBC), L2 cache/SRAM) **Note:** If TGINT is set to an I/O port (for example, PCI/PCI-X) 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 target of the transaction (to I/O interface or to local memory). Following are the transaction type settings for reads from the PCI/PCI-X interface: 0000–0011 Reserved 0100 Read 0101–1111 Reserved The following are the transaction type settings for reads from local memory: 0000–0011 Reserved 0100 Received read; do not snoop processor core (e500) 0101 Received read; snoop processor core (e500) 0110 Reserved 0111 Received read; unlock L2 cache line 1000–1111 Reserved
16–19	WRTYP	Write transaction type. Transaction type to run if access is a write. The field description depends on the target of the transaction (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 processor core (e500) 0101 Received write; snoop processor core (e500) 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 Kbytes 00_1100 8 Kbytes ... 01_1111 4 Gbytes 10_0000–11_1111 Reserved

16.3.2.3 Error Management Registers

RapidIO error management registers allow configuration of RapidIO error detection, capture, and reporting. Bits in the port notification/fatal error detect register (PNFEDR) report which port notification or error was detected, subject to individual detection enable bit settings in the port notification/fatal error detect disable register (PNFEDiR). Upon detecting a port notification or

error, an interrupt is generated subject to the interrupt enable bit settings in the port notification/fatal error interrupt enable register (PNFEIER); the port error capture status register (PECSR) indicates whether a valid packet was captured when the port notification or fatal error occurred. If a packet is captured (PECSR[V] is set), packet information may be found in the error packet capture registers (EPCR*n*). The packet format type, as captured in EPCR0[PFT], determines the format of the remaining packet data captured in EPCR1 and EPCR2.

Bits in the port recoverable error detect register (PREDR) report which recoverable errors have been detected.

The port error recovery threshold register (PERTR) and the port retry threshold register (PRTR) allow software to assign and monitor the recovery and retry thresholds, which are referenced by hardware before reporting a recovery error (PNFEDR[ETE]) or a retry error (PNFEDR[RTE]).

16.3.2.3.1 Port Notification/Fatal Error Detect Register (PNFEDR)

The PNFEDR, shown in Figure 16-39, reflects individual port notifications and fatal errors that the RapidIO controller has detected. Each bit is cleared when a 1 is written to it.

	0	1	2	3		5	6	7	8	9	10	11	12	13	14	15
R	MFE	IAXE	OAXE	0	0	0	ITE	MSE	NTIE	SE	DSE	MLE	ETE	RTE	RERE	LTE
W																
Reset	0000_0000_0000_0000															

	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	0000_0000_0000_0000															
Offset	0xD_0E00															

Figure 16-39. Port Notification/Fatal Error Detect Register (PNFEDR)

Table 16-41 describes the fields of the PNFEDR.

Table 16-41. PNFEDR Field Descriptions

Bits	Name	Description
0	MFE	Multiple fatal errors. (Bit reset, write-one-to-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	IAXE	Inbound ATMU window crossing error. (Bit reset, write-one-to-clear) 0 No inbound window crossing error 1 This error indicates that an inbound transaction either crosses the boundary of its ATMU window or crosses into a higher priority ATMU window.

Table 16-41. PNFEDR Field Descriptions (continued)

Bits	Name	Description
2	OAXE	Outbound ATMU window crossing error. (Bit reset, write-one-to-clear) 0 No outbound window crossing error 1 This error indicates that an outbound transaction either crosses the boundary of its ATMU window or crosses into a higher priority ATMU window.
3–5	—	Reserved
6	ITE	Illegal transaction target error. (Bit reset, write-one-to-clear) 0 No illegal transaction detected 1 Received a packet whose target ID does not match the RapidIO controller deviceID determined at POR while accept_all mode is disabled
7	MSE	Message size error. (Bit reset, write-one-to-clear) 0 No message size error detected 1 Received message packet data payload is not of the size specified in the ssize field (with the exception of the last packet which may be less)
8	NTIE	Missing idle after training. (Bit reset, write oneto 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-one-to-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-one-to-clear) 0 No duplicate message segment received 1 Received a duplicate message segment from RapidIO
11	MLE	Message length error. (Bit reset, write-one-to-clear) 0 No message length error detected 1 Received a message packet with the same mailbox, letter, and source ID as the current outstanding (not completed) message, but with a different message length.
12	ETE	Error recovery threshold error. (Bit reset, write-one-to-clear) 0 No error recovery threshold error detected 1 Error recovery threshold count (defined in PERTR[RCTT]) exceeded
13	RTE	Retry threshold error. (Bit reset, write-one-to-clear) 0 No retry threshold error detected 1 Consecutive ack retry control symbols received threshold count (defined in PRTR[RTT]) exceeded
14	RERE	Received request ERROR response. (Bit reset, write-one-to-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-one-to-clear) 0 No link response time-out 1 A link response is not received within the time-out interval, as defined by PLTOCCSR[TV].
16	RSTE	Packet response time-out. (Bit reset, write-one-to-clear) 0 No packet response time-out 1 A packet response is not received within the time-out interval, as defined by PRTOCCSR[TV].
17	NAE	Nonsensical ackID. (Bit reset, write-one-to-clear) 0 AckID OK 1 Unrecognized ackID received with link response

Table 16-41. PNFEDR Field Descriptions (continued)

Bits	Name	Description
18	IRFE	Illegal request fields 0 Request packet fields OK 1 Illegal combinations of fields in the request packet. (Bit reset, write-one-to-clear)
19	URDE	Unexpected request data 0 Request data OK 1 Read type with a data payload. (Bit reset, write-one-to-clear).
20	URE	Unsupported request 0 Packet format type OK 1 Unsupported packet format type or transport type. (Bit reset, write-one-to-clear)
21	URSE	Unexpected request size 0 Packet data size OK 1 Packet data is not expected size with respect to the packet size bits (it reset, write-one-to-clear)
22	IWRE	Illegal write request 0 Write request OK 1 Write type with no data payload. (Bit reset, write-one-to-clear)
23	IRSFE	Illegal response fields 0 Response packet OK 1 Illegal combinations of fields in the response packet. (Bit reset, write-one-to-clear)
24	IRTE	Illegal response type 0 Response type OK 1 Illegal response for a given request type. (Bit reset, write-one-to-clear)
25	BRRE	Bad read response 0 Read response OK 1 Read data response with no data payload. (Bit reset, write-one-to-clear)
26	BWRE	Bad write response 0 Write response OK 1 Write response with a data payload. (Bit reset, write-one-to-clear)
27	URSSE	Unexpected response data size 0 Response packet size OK 1 Response packet data is not expected size. (Bit reset, write-one-to-clear)
28	URESE	Unsolicited response without data 0 No response without data condition 1 A response without data is received if there exists no outstanding request matching that TID. (Bit reset, write-one-to-clear).
29	URSDE	Unsolicited response with data 0 No unsolicited response with data condition 1 A response with data is received if there exists no outstanding request matching that TID. (Bit reset, write-one-to-clear)
30	MERE	Message error response 0 No message error response 1 Received a response of type error for an outbound message packet. (Bit reset, write-one-to-clear)
31	MDFE	Message descriptor fetch error 0 No message descriptor fetch error 1 A message descriptor fetch from local memory results in an error. (Bit reset, write-one-to-clear)

16.3.2.3.2 Port Notification/Fatal Error Detect Disable Register (PNFEDiR)

The PNFEDiR individually disables the detection of each port notification or fatal error defined in the PNFEDR. Figure 16-40 shows the PNFEDiR.

	0	1	2	3		5	6	7	8	9	10	11	12	13	14	15
R	0	IAXD	OAXD	0	0	0	ITD	MSD	NTID	SD	DSD	BMD	ETD	RTD	RERD	LTD
W																
Reset	0000_0000_0000_0000															

	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	BWRD	URSSD	URESD	URSDD	MERD	MDFD
W																
Reset	0000_0000_0000_0000															
Offset	0xD_0E04															

Figure 16-40. Port Notification/Fatal Error Detect Disable Register (PNFEDiR)

Table 16-42 describes the fields of the PNFEDiR.

Table 16-42. PNFEDiR Field Descriptions

Bits	Name	Description
0	—	Reserved
1	IAXD	Inbound ATMU window crossing error detection disable 0 Error detection enabled 1 Error detection disabled
2	OAXD	Outbound ATMU window crossing error detection disable 0 Error detection enabled 1 Error detection disabled
3–5	—	Reserved
6	ITD	Illegal transaction target error detection disable 0 Error detection enabled 1 Error detection disabled
7	MSD	Message size error detection disable 0 Error detection enabled 1 Error detection disabled
8	NTID	Missing idle after training error detection disable 0 Error detection enabled 1 Error detection disabled
9	SD	Message segment error detection disable 0 Error detection enabled 1 Error disabled

Table 16-42. PNFEDiR Field Descriptions (continued)

Bits	Name	Description
10	DSD	Duplicate message segment error detection disable 0 Error detection enabled 1 Error detection disabled
11	BMD	Message length error detection disable 0 Error detection enabled 1 Error detection disabled
12	ETD	Error recovery threshold error detection disable 0 Error detection enabled 1 Error detection disabled
13	RTD	Retry threshold error detection disable 0 Error detection enabled 1 Error detection disabled
14	RERD	Received ERROR response error detection disable 0 Error detection enabled 1 Error detection disabled
15	LTD	Link response time-out error detection disable 0 Error detection enabled 1 Error detection disabled
16	RSTD	Packet response time-out error detection disable 0 Error detection enabled 1 Error detection disabled
17	NAD	Nonsensical ackID error detection disable 0 Error detection enabled 1 Error detection disabled
18	IRFD	Illegal request fields error detection disable 0 Error detection enabled 1 Error detection disabled
19	URDD	Unexpected request data error detection disable 0 Error detection enabled 1 Error detection disabled
20	URD	Unsupported request error detection disable 0 Error detection enabled 1 Error detection disabled
21	URSD	Unexpected request size error detection disable 0 Error detection enabled 1 Error detection disabled
22	IWRD	Illegal write request error detection disable 0 Error detection enabled 1 Error detection disabled
23	IRSFD	Illegal response fields error detection disable 0 Error detection enabled 1 Error detection disabled
24	IRTD	Illegal response type error detection disable 0 Error detection enabled 1 Error detection disabled

Table 16-42. PNFEDiR Field Descriptions (continued)

Bits	Name	Description
25	BRRD	Bad read response error detection disable 0 Error detection enabled 1 Error detection disabled
26	BWRD	Bad write response error detection disable 0 Error detection enabled 1 Error detection disabled
27	URSSD	Unexpected response data size error detection disable 0 Error detection enabled 1 Error detection disabled
28	URESD	Unsolicited response without data error detection disable 0 Error detection enabled 1 Error detection disabled
29	URSDD	Unsolicited response with data error detection disable 0 Error detection enabled 1 Error detection disabled
30	MERD	Message error response error detection disable 0 Error detection enabled 1 Error detection disabled
31	MDFD	Message descriptor fetch error detection disable 0 Error detection enabled 1 Error detection disabled

16.3.2.3.3 Port Notification/Fatal Error Interrupt Enable Register (PNFEIER)

The PNFEIER, shown in Figure 16-41, individually enables each detected port notification or fatal error to generate an interrupt to the programmable interrupt controller (PIC).

	0	1	2	3		5	6	7	8	9	10	11	12	13	14	15
R	0	IAXIE	OAXIE	0	0	0	ITIE	MSIE	NTIIE	SIE	DSIE	BMIE	ETIE	RTIE	RERIE	LTIE
W																
Reset	0000_0000_0000_0000															

	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	URESIE	URSDIE	MERIE	MDFIE
W																
Reset	0000_0000_0000_0000															
Offset	0xD_0E08															

Figure 16-41. Port Notification/Fatal Error Interrupt Enable Register (PNFEIER)

Table 16-43 describes the fields of the PNFEIER.

Table 16-43. PNFEIER Field Descriptions

Bits	Name	Description
0	—	Reserved
1	IAXIE	Inbound ATMU window crossing error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
2	OAXIE	Outbound ATMU window crossing error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
3–5	—	Reserved
6	ITIE	Illegal transaction target error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
7	MSIE	Message size error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
8	NTIIE	Missing idle after training interrupt enable 0 Interrupt enabled 1 Interrupt disabled
9	SIE	Message segment error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
10	DSIE	Duplicate message segment error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
11	BMIE	Message length error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
12	ETIE	Error recovery threshold error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
13	RTIE	Retry threshold error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
14	RERIE	Received ERROR response error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
15	LTIE	Link response time-out error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
16	RSTIE	Packet response time-out error interrupt enable 0 Interrupt enabled 1 Interrupt disabled

Table 16-43. PNFEIER Field Descriptions (continued)

Bits	Name	Description
17	NAIE	Nonsensical ackID error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
18	IRFIE	Illegal request fields error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
19	URDIE	Unexpected request data error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
20	URIE	Unsupported request error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
21	URSIE	Unexpected request size error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
22	IWRIE	Illegal write request error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
23	IRSFIE	Illegal response fields error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
24	IRTIE	Illegal response type error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
25	BRRIE	Bad read response error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
26	BWRIE	Bad write response error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
27	URSSIE	Unexpected response data size error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
28	URESIE	Unsolicited response without data error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
29	URSDIE	Unsolicited response with data error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
30	MERIE	Message error response error interrupt enable 0 Interrupt enabled 1 Interrupt disabled
31	MDFIE	Message descriptor fetch error interrupt enable 0 Interrupt enabled 1 Interrupt disabled

16.3.2.3.4 Port Error Capture Status Register (PECSR)

The PECSR, shown in Figure 16-42, indicates whether a valid packet has been captured when a port notification or fatal error has occurred. Hardware sets PECSR[V] and software is expected to clear it after processing the error to capture future error packets.

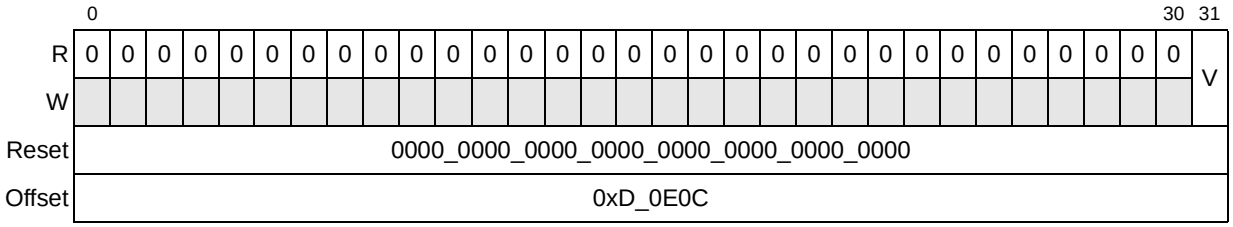


Figure 16-42. Port Error Capture Status Register (PECSR)

Table 16-44 describes the fields of the PECSR.

Table 16-44. PECSR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	V	Packet register information valid 0 Packet capture registers do not contain valid information. 1 Packet capture registers contain valid information.

16.3.2.3.5 Error Packet Capture Register 0 (EPCR0)

The EPCR0, shown in Figure 16-43, captures the first 32 bits of an inbound packet that caused a port notification or fatal error. The format of the first 32 bits of an inbound packet are the same for all packet format types.

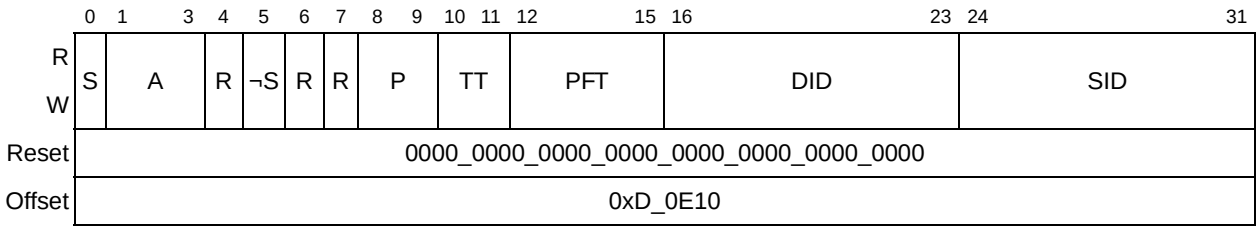


Figure 16-43. Error Packet Capture Register 0 (EPCR0)

Table 16-45 describes the fields of the EPCR0.

Table 16-45. EPCR0 Field Descriptions

Bits	Name	Description
0	S	S-bit of the packet
1–3	A	AckID field of a packet
4	R	These bits correspond to bits 4–7 of a packet header. In the packet definition bit 4 is a reserved packet field, bit 5 is S-complement, and bits 6 and 7 are reserved packet fields. (Bits 4, 6 and 7 are reserved packet fields, not reserved register fields.)
5	¬S	
6–7	R	
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

16.3.2.3.6 Error Packet Capture Register 1 (EPCR1)

EPCR1 captures a portion of an inbound packet that caused a port notification or a fatal error. The format of the EPCR1 contents vary subject to the packet format type as captured in EPCR0[PFT].

The various content formats of EPCR1 are described in Section 16.3.2.3.7, “EPCR1—Type 1 Packet Format,” through Section 16.3.2.3.15, “EPCR1—Type 13 Packet Format.”

16.3.2.3.7 EPCR1—Type 1 Packet Format

Figure 16-44 describes EPCR1 for a type 1 packet format (EPCR0[PFT] = 1).

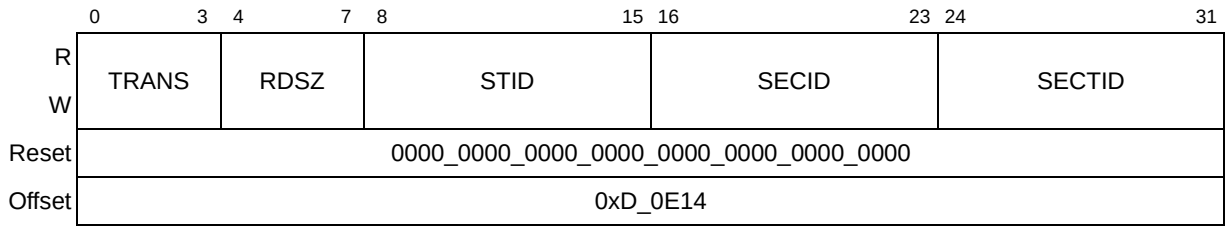


Figure 16-44. Error Packet Capture Register 1 (EPCR1)—Type 1 Packet Format

Table 16-46 describes the EPCR1 fields for a type 1 packet format (EPCR0[PFT] = 1).

Table 16-46. EPCR1 Type 1 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

16.3.2.3.8 EPCR1—Type 2 Packet Format

Figure 16-45 describes the EPCR1 for a type 2 packet format (EPCR0[PFT] = 2).

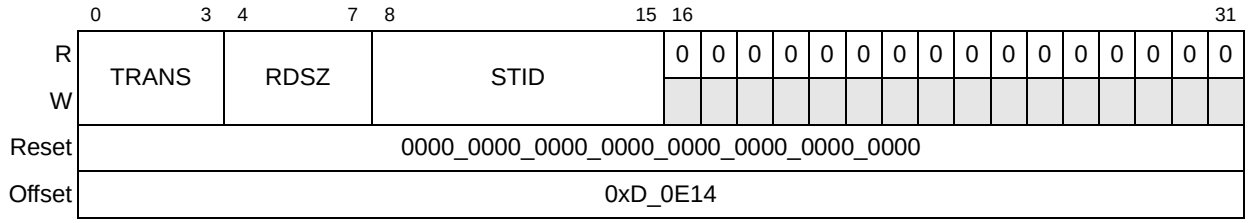


Figure 16-45. Error Packet Capture Register 1 (EPCR1)—Type 2 Packet Format

Table 16-47 describes the EPCR1 fields for a type 2 packet format (EPCR0[PFT] = 2).

Table 16-47. EPCR1 Type 2 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

16.3.2.3.9 EPCR1—Type 5 Packet Format

Figure 16-46 describes the EPCR1 for a type 5 packet format (EPCR0[PFT] = 5).

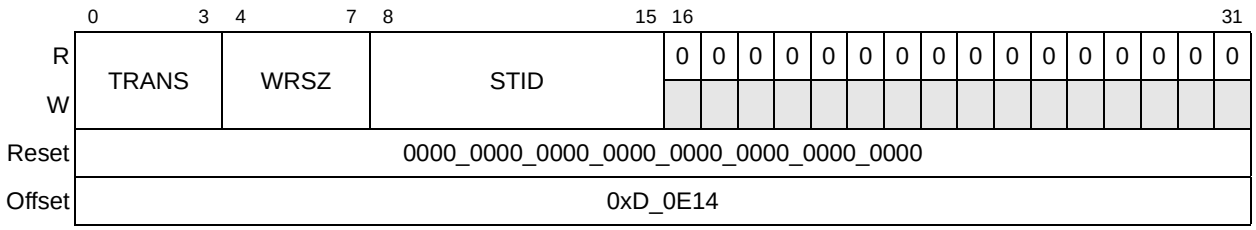


Figure 16-46. Error Packet Capture Register 1 (EPCR1)—Type 5 Packet Format

Table 16-48 describes the EPCR1 fields for a type 5 packet format (EPCR0[PFT] = 5).

Table 16-48. EPCR1 Type 5 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

16.3.2.3.10 EPCR1—Type 6 Packet Format

Figure 16-47 shows the EPCR1 for a type 6 packet format (EPCR0[PFT] = 6).

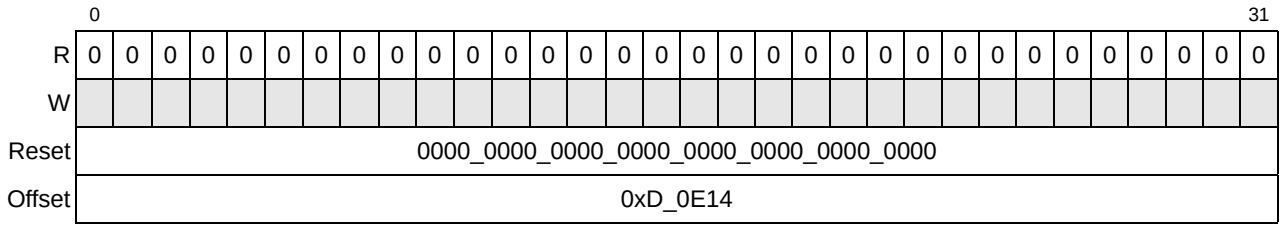


Figure 16-47. Error Packet Capture Register 1 (EPCR1)—Type 6 Packet Format

Table 16-49 describes the field of the EPCR1 for a type 6 packet format (EPCR0[PFT] = 6).

Table 16-49. EPCR1 Type 6 Field Descriptions

Bits	Name	Description
0–31	—	Reserved

16.3.2.3.11 EPCR1—Type 8 Request Packet Format

Figure 16-48 shows EPCR1 for a type 8 request packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 0 or 1).

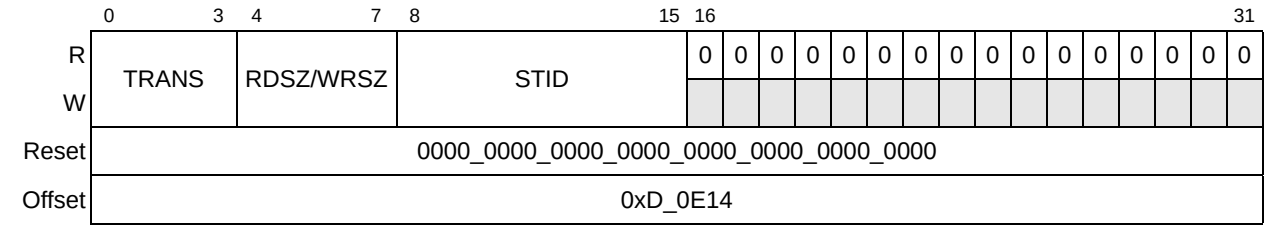


Figure 16-48. Error Packet Capture Register 1 (EPCR1)—Type 8 Request Packet Format

Table 16-50 describes EPCR1 fields for a type 8 request packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 0 or 1).

Table 16-50. EPCR1 Type 8 Request 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

16.3.2.3.12 EPCR1—Type 8 Response Packet Format

Figure 16-49 describes the EPCR1 for a type 8 response packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 2 or 3).

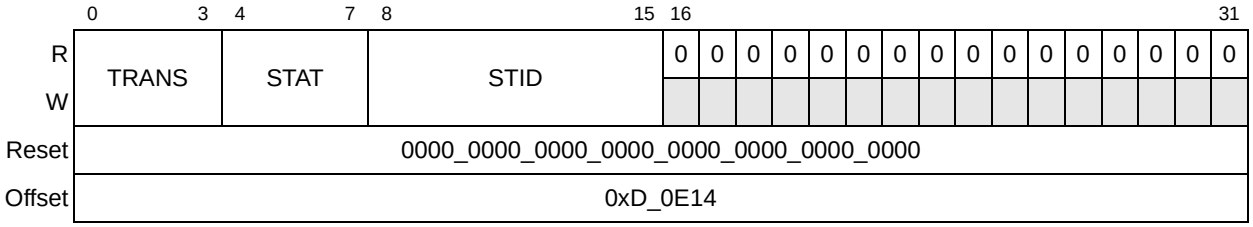


Figure 16-49. Error Packet Capture Register 1 (EPCR1)—Type 8 Response Packet Format

Table 16-51 describes the fields of the EPCR1 for a type 8 response packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 2 or 3).

Table 16-51. EPCR1 Type 8 Response 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

16.3.2.3.13 EPCR1—Type 10 Packet Format

Figure 16-50 describes the EPCR1 for a type 10 packet format (EPCR0[PFT] = 10).

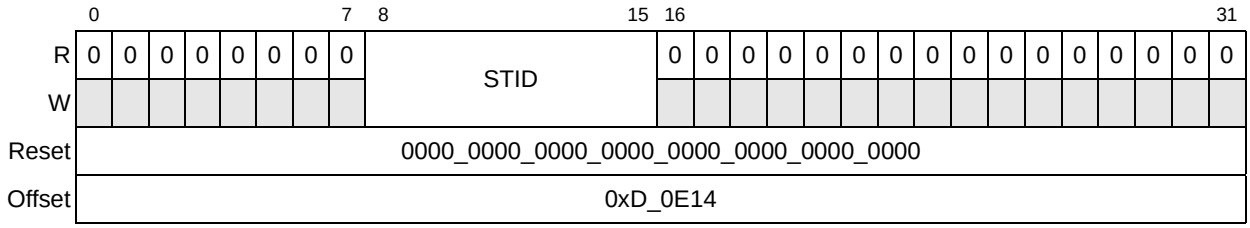


Figure 16-50. Error Packet Capture Register 1 (EPCR1)—Type 10 Packet Format

Table 16-52 describes EPCR1 fields for a type 10 packet format (EPCR0[PFT] = 10).

Table 16-52. EPCR1 Type 10 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	STID	Source transaction ID
16–31	—	Reserved

16.3.2.3.14 EPCR1—Type 11 Packet Format

Figure 16-51 describes the EPCR1 for a type 11 packet format (EPCR0[PFT] = 11).

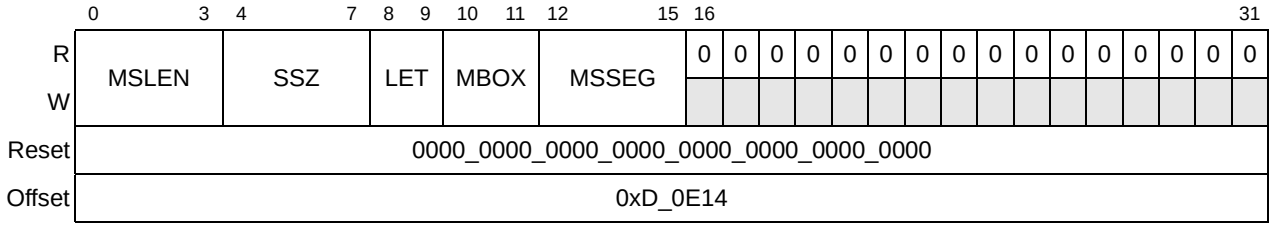


Figure 16-51. Error Packet Capture Register 1 (EPCR1)—Type 11 Packet Format

Table 16-53 describes EPCR1 fields for a type 11 packet format (EPCR0[PFT] = 11).

Table 16-53. EPCR1 Type 11 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

16.3.2.3.15 EPCR1—Type 13 Packet Format

Figure 16-52 describes the EPCR1 for a type 13 packet format (EPCR0[PFT] = 13).

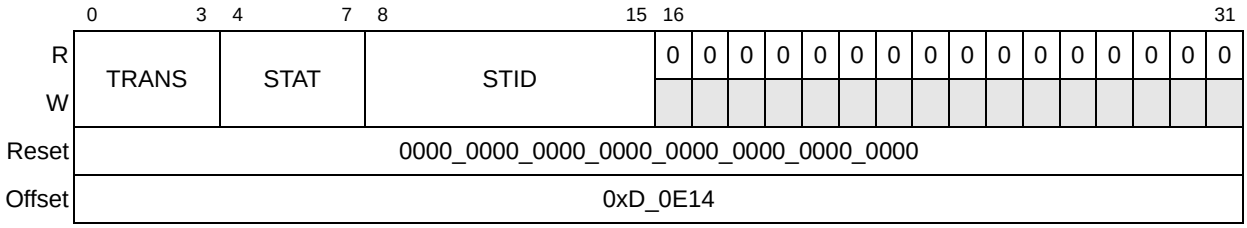


Figure 16-52. Error Packet Capture Register 1 (EPCR1)—Type 13 Packet Format

Table 16-54 describes EPCR1 fields for a type 13 packet format (EPCR0[PFT] = 13).

Table 16-54. EPCR1 Type 13 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

16.3.2.3.16 Error Packet Capture Register 2 (EPCR2)

This register captures another portion of an inbound packet that caused a port notification or a fatal error. The format of the contents of this register are vary subject to the packet format type as captured in EPCR0[PFT]. The various content formats of this register are described in Section 16.3.2.3.17, “EPCR2—Type 1, 2, or 5 Packet Format,” through Section 16.3.2.3.22, “EPCR2—Type 11 or 13 Packet Format.”

16.3.2.3.17 EPCR2—Type 1, 2, or 5 Packet Format

Figure 16-53 shows EPCR2 for a type 1, 2, or 5 packet format (EPCR0[PFT] = 1, 2, or 5).

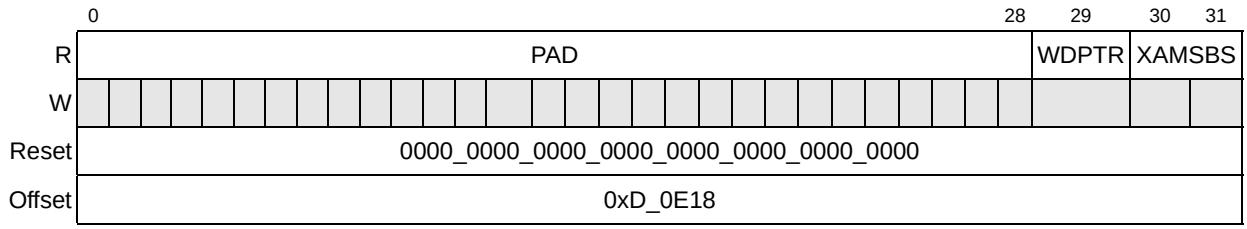


Figure 16-53. Error Packet Capture Register 2 (EPCR2)—Type 1, 2, or 5 Packet Format

Table 16-55 shows EPCR2 for a type 1, 2, or 5 packet format (EPCR0[PFT] = 1, 2, or 5).

Table 16-55. EPCR2 Type 1, 2, or 5 Field Descriptions

Bits	Name	Description
0–28	PAD	Address[2:30] field of a packet
29	WDPTR	Word pointer field of a packet, which is used in conjunction with the data size field
30–31	XAMSBS	Address[0:1] field of a packet

16.3.2.3.18 EPCR2—Type 6 Packet Format

Figure 16-54 describes the EPCR2 for a type 6 packet format (EPCR0[PFT] = 6).

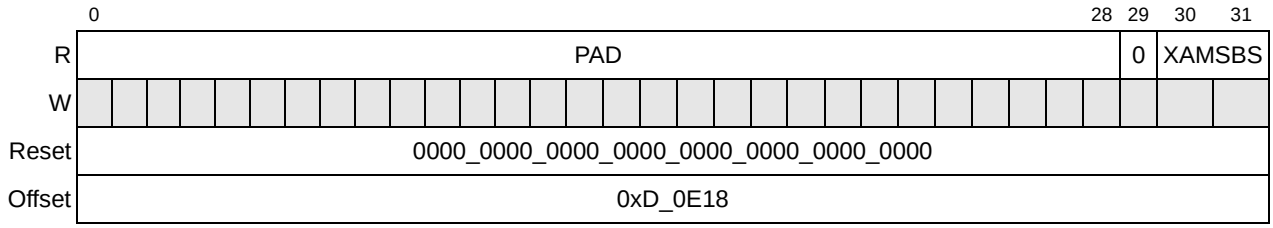


Figure 16-54. Error Packet Capture Register 2 (EPCR2)—Type 6 Packet Format

Table 16-56 describes the EPCR2 fields for a type 6 packet format (EPCR0[PFT] = 6).

Table 16-56. EPCR2 Type 6 Field Descriptions

Bits	Name	Description
0–28	PAD	Address[2:30] field of a packet
29	—	Reserved
30–31	XAMSBS	Address[0:1] field of a packet

16.3.2.3.19 EPCR2—Type 8 Request Packet Format

Figure 16-55 shows EPCR2 for a type 8 request packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 0 or 1).

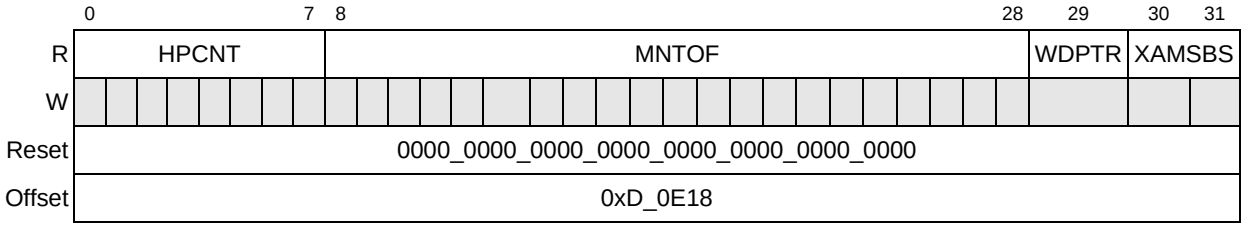


Figure 16-55. Error Packet Capture Register 2 (EPCR2)—Type 8 Request Packet Format

Table 16-57 describes the fields of the EPCR2 for a type 8 request packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 0 or 1).

Table 16-57. EPCR2 Type 8 Request 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

16.3.2.3.20 EPCR2—Type 8 Response Packet Format

Figure 16-56 shows EPCR2 for a type 8 response packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 2 or 3).

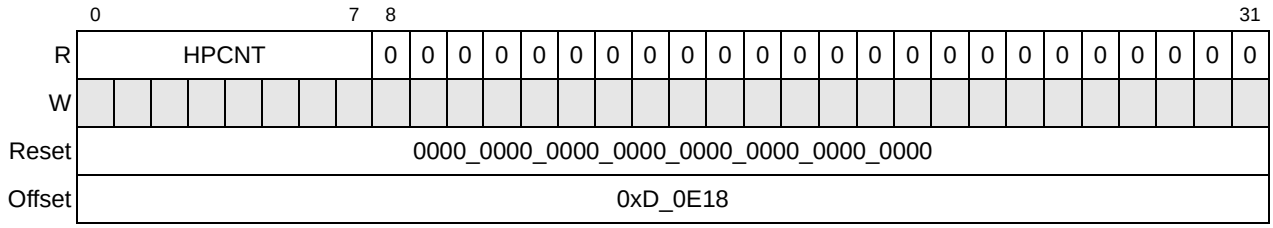


Figure 16-56. Error Packet Capture Register 2 (EPCR2)—Type 8 Response Packet Format

Table 16-58 describes EPCR2 for a type 8 response packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 2 or 3).

Table 16-58. EPCR2 Type 8 Response Field Descriptions

Bits	Name	Description
0–7	HPCNT	Hop count field of a packet
8–31	—	Reserved

16.3.2.3.21 EPCR2—Type 10 Packet Format

Figure 16-57 describes the EPCR2 for a type 10 packet format (EPCR0[PFT] = 10).

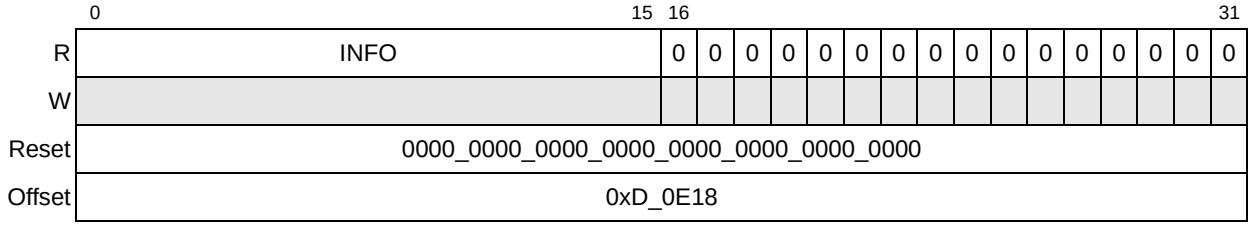


Figure 16-57. Error Packet Capture Register 2 (EPCR2)—Type 10 Packet Format

Table 16-59 describes EPCR2 for a type 10 packet format (EPCR0[PFT] = 10).

Table 16-59. EPCR2 Type 10 Field Descriptions

Bits	Name	Description
0–15	INFO	Information field of a packet
16–31	—	Reserved

16.3.2.3.22 EPCR2—Type 11 or 13 Packet Format

Figure 16-58 shows EPCR2 for a type 11 or 13 packet format (EPCR0[PFT] = 11 or 13).

	0																													31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																														
Reset	0000_0000_0000_0000_0000_0000_0000_0000																													
Offset	0xD_0E18																													

Figure 16-58. Error Packet Capture Register 2 (EPCR2)—Type 11 or 13 Packet Format

Table 16-60 describes EPCR2 for a type 11 or 13 packet format (EPCR0[PFT] = 11 or 13).

Table 16-60. EPCR2 Type 11 or 13 Field Description

Bits	Name	Description
0–31	—	Reserved

16.3.2.3.23 Port Recoverable Error Detect Register (PREDR)

The PREDR, described in Figure 16-59, 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	IE256E	INAIE	INAUA
W																
Reset	0000_0000_0000_0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INACC	INAPS	INABC	INASE	INAGE	URE	UAE	ABRE	UTE	USE	ULE	UEE	LRE	FAE	FTEE	ATE
W																
Reset	0000_0000_0000_0000															
Offset	0xD_0E20															

Figure 16-59. Port Recoverable Error Detect Register (PREDR)

Table 16-61 describes the fields of the PREDR.

Table 16-61. PREDR Field Descriptions

Bits	Name	Description
0	MRE	Multiple recoverable errors. (Bit reset, write-one-to-clear) 0 Multiple recoverable RapidIO errors of the same type were not detected. 1 Multiple recoverable RapidIO errors of the same type were detected.
1–6	—	Reserved
7	ETE	Embedded training. (Bit reset, write-one-to-clear) 0 No requested training pattern 1 Requested training pattern is embedded in a packet
8	ISE	Inbound S-bit error. (Bit reset, write-one-to-clear) 0 No inbound S-bit error 1 Received a packet/control symbol with an S-bit parity error
9	ICCE	Inbound corrupt control symbol. (Bit reset, write-one-to-clear) 0 No inbound corrupt control symbol 1 Received a control symbol in which true does not match complement counterpart
10	IAOSE	Inbound out-of-sequence ack symbol. (Bit reset, write-one-to-clear) 0 No inbound out-of-sequence ack symbol 1 Received an ack control symbol with an unexpected ackID
11	IUAE	Inbound packet with unexpected ackID. (Bit reset, write-one-to-clear) 0 No inbound packet with unexpected ackID 1 Received packet with unexpected ackID value—out-of-sequence ackID
12	IBCE	Inbound packet with bad CRC. (Bit reset, write-one-to-clear) 0 No inbound packet with bad CRC 1 Received packet with a bad CRC value
13	IE256E	Inbound packet exceeds 256 bytes. (Bit reset, write-one-to-clear) 0 The data payload of an inbound packet does not exceed 256 bytes. 1 The data payload of an inbound packet exceeds 256 bytes.
14	INAIE	Packet-not-accepted; encountered internal error. (Bit reset, write-one-to-clear) 0 No packet-not-accepted; internal error encountered 1 Received ack control symbol with packet-not-accepted of type ‘encountered internal error’
15	INAUA	Packet-not-accepted; unexpected ackID on packet. (Bit reset, write-one-to-clear) 0 No packet-not-accepted; unexpected ackID on packet condition 1 Received ack control symbol with packet-not-accepted of type ‘unexpected ackID’
16	INACC	Packet-not-accepted; corrupt control symbol. (Bit reset, write-one-to-clear) 0 No packet-not-accepted; corrupt control symbol condition 1 Received ack control symbol with packet-not-accepted of type ‘corrupt control symbol’
17	INAPS	Packet-not-accepted; input port stopped. (Bit reset, write-one-to-clear) 0 No packet-not-accepted; input port stopped condition 1 Received ack control symbol with packet-not-accepted of type ‘input port stopped’
18	INABC	Packet-not-accepted; bad CRC. (Bit reset, write-one-to-clear) 0 No packet-not-accepted; bad CRC condition 1 Received ack control symbol with packet-not-accepted of type ‘bad CRC’

Table 16-61. PREDR Field Descriptions (continued)

Bits	Name	Description
19	INASE	Packet-not-accepted; S-bit parity error. (Bit reset, write-one-to-clear) 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’
20	INAGE	Packet-not-accepted; general error. (Bit reset, write-one-to-clear) 0 No packet-not-accepted; general error condition 1 Received ack control symbol with packet-not-accepted of type ‘general error’
21	URE	Unexpected restart-from-retry symbol. (Bit reset, write-one-to-clear) 0 No unexpected restart-from-retry symbol 1 Received a restart-from-retry control symbol while in the OK state
22	UAE	Unsolicited ACK symbol. (Bit reset, write-one-to-clear) 0 No unsolicited ack symbol encountered 1 Received an ack control symbol while no packets are posted on the outstanding ack history queue
23	ABRE	Ack before restart-from-retry. (Bit reset, write-one-to-clear) 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
24	UTE	Unexpected training symbol. (Bit reset, write-one-to-clear) 0 No unexpected training symbol 1 Received a training symbol while in the OK state
25	USE	Unexpected stomp symbol. (Bit reset, write-one-to-clear) 0 No unexpected stomp symbol 1 Received a stomp control symbol while there is no packet being received
26	ULE	Unexpected link response symbol. (Bit reset, write-one-to-clear) 0 No unexpected link response symbol 1 Received a link response control symbol while there is no outstanding request
27	UEE	Unexpected EOP symbol. (Bit reset, write-one-to-clear) 0 No unexpected EOP symbol 1 Received an EOP control symbol while there is no packet being received
28	LRE	Link request error. (Bit reset, write-one-to-clear) 0 No link request error 1 Received a link request control symbol before the previous link request has been serviced
29	FAE	Frame toggle alignment. (Bit reset, write-one-to-clear) 0 No frame toggle alignment problem 1 Received FRAME signal toggles at a non 32-bit boundary
30	FTEE	Frame toggle edge. (Bit reset, write-one-to-clear) 0 No frame toggle edge problem 1 Received FRAME signal toggles on the negative edge of the clock
31	ATE	Ack time-out. (Bit reset, write-one-to-clear) 0 No ack time-out condition 1 An ack control symbol is not received within the specified time-out interval.

16.3.2.3.24 Port Error Recovery Threshold Register (PERTR)

Figure 16-60 shows the PERTR.

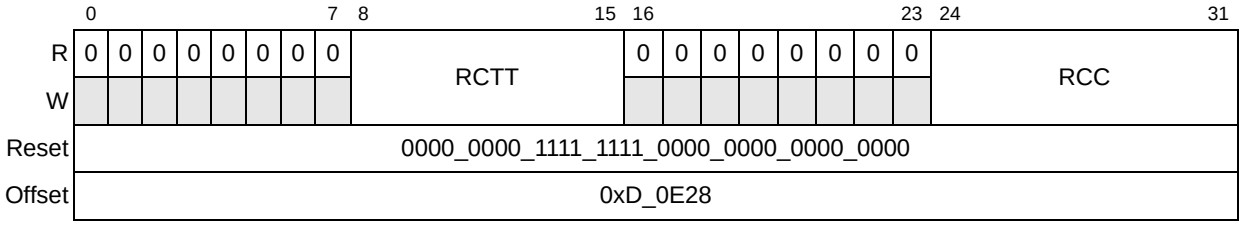


Figure 16-60. Port Error Recovery Threshold Register (PERTR)

Table 16-62 describes the fields of the PERTR.

Table 16-62. 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. Maintains a count of the number of times the port has successfully recovered from a transmission error. An error is reported (PNFEDR[ETE] is set) if this value equals the value contained in the recovery threshold trigger field and if reporting is enabled (PNFEDiR[ETD] is cleared).

16.3.2.3.25 Port Retry Threshold Register (PRTR)

Figure 16-61 shows the PRTR.

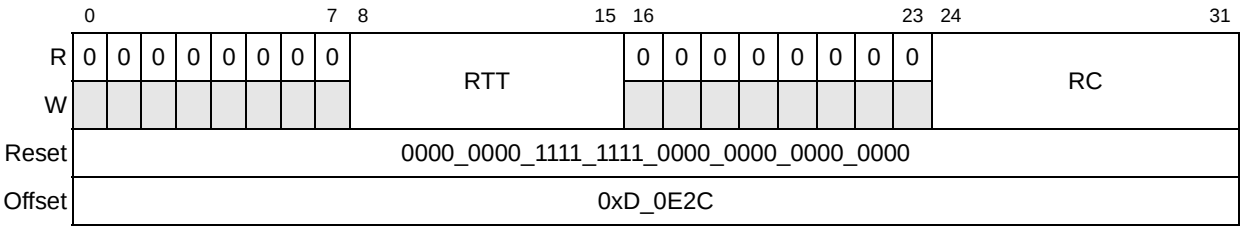


Figure 16-61. Port Retry Threshold Register (PRTR)

Table 16-63 describes the fields of the PRTR.

Table 16-63. PRTR Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	RTT	Retry threshold trigger. These bits provide the threshold value for the number of consecutive ack retries received for an outbound packet before reporting an error condition.
16–23	—	Reserved
24–31	RC	Retry counter. Maintains a count of the number of consecutive times the port has received an ack retry for an outbound packet. An error is reported (PNFEDR[RTE] is set) if this value equals the value contained in the retry threshold trigger field and if reporting is enabled (PNFEDiR[RTD] is cleared).

16.3.3 RapidIO Message Unit Registers

This section describes RapidIO outbound and inbound message registers, doorbell registers and port-write registers. Refer to Section 16.8, “RapidIO Message Unit,” for additional information about how these registers affect the message unit.

16.3.3.1 RapidIO Outbound Message Registers

The registers in this section control RapidIO outbound messages. The following sections provide descriptions of these registers. Additional information may be found in the *Parallel RapidIO Intconnect Specification*.

16.3.3.1.1 Outbound Mode Register (OMR)

The outbound mode register (OMR) allows software to start a message operation and to control various message operation characteristics. Figure 16-62 shows the OMR.

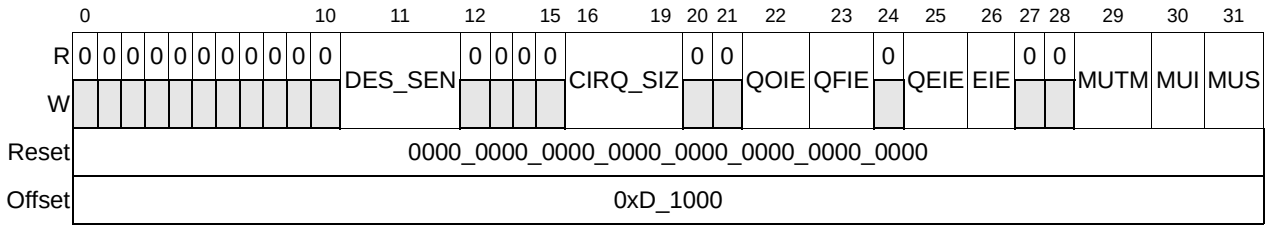


Figure 16-62. Outbound Mode Register (OMR)

Table 16-64 describes the fields of the OMR.

Table 16-64. OMR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	DES_SEN	Descriptor snoop enable 0 Snooping of processor core disabled 1 Snooping of the processor core when reading descriptors from memory enabled
12–15	—	Reserved
16–19	CIRQ_SIZ	Circular descriptor queue size. Determines the number of descriptors that can be placed in 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 1011–1111 Reserved
20–21	—	Reserved
22	QOIE	Queue overflow interrupt enable. Only applicable to chaining mode 0 No QO interrupt 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 software and the queue was full)
23	QFIE	Queue full interrupt enable. Only applicable to chaining mode. 0 No QF interrupt generated 1 Generates an interrupt when the queue transitions to full (that is, the enqueue and dequeue pointers are equal after being incremented by software)
24	—	Reserved
25	QEIE	Queue empty interrupt enable. Only applicable to chaining mode 0 No QE interrupt 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	EIE	Error interrupt enable 0 No interrupt is generated if a programming or transfer error is detected. 1 Generates an interrupt if a programming or transfer error is detected
27–28	—	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.

Table 16-64. OMR Field Descriptions (continued)

Bits	Name	Description
30	MUI	Message unit increment. Software sets MUI after writing a descriptor to memory. Hardware then increments the ODQEPAR and clears this bit.Always reads as 0 if MUS is set
31	MUS	Message unit start • Direct mode: a 0 to 1 transition while the message unit is not busy (OSR[MUB] = 0) 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 (OSR[MUB] = 0) 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 MUS 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 this halt condition.

16.3.3.1.2 Outbound Status Register (OSR)

The outbound status register (OSR), shown in Figure 16-63, reports various message unit conditions during and after a message operation. Writing a 1 to a writable condition bit in OSR clears it.

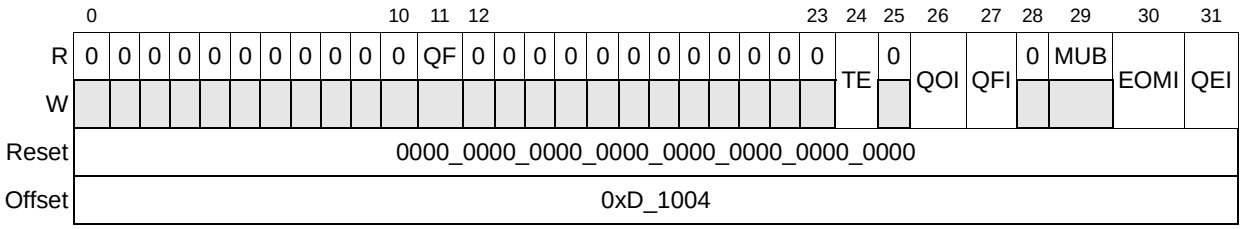


Figure 16-63. Outbound Status Register (OSR)

Table 16-65 describes the fields of the OSR.

Table 16-65. OSR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	QF	Queue full (read-only) 0 Queue not full 1 The queue has become full.
12–23	—	Reserved
24	TE	Transaction error 0 No transaction error 1 There was an error condition during the message operation. The message unit has received a response error and halts. An interrupt is generated if OMR[IEI] is set (Bit reset, write-one-to-clear, resuming message unit operation)
25	—	Reserved

Table 16-65. OSR Field Descriptions (continued)

Bits	Name	Description
26	QOI	Queue overflow interrupt. Only applicable to chaining mode. (Bit reset, write-one-to-clear) 0 No overflow interrupt condition 1 Queue overflow 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-one-to-clear) 0 No queue full interrupt 1 The queue became full and OMR[QFIE] is set. An interrupt is generated in this case.
28	—	Reserved
29	MUB	Message unit busy (read-only) 0 Message unit not busy. Cleared when the message unit is no longer busy 1 A message operation is currently in progress.
30	EOMI	End-of-message interrupt. (Bit reset, write-one-to-clear) 0 No end-of-message interrupt condition 1 After finishing this message operation, if ODATR[EOMIE] is set, EOMI is set and an interrupt is generated
31	QEI	Queue empty interrupt. Only applicable to chaining mode. (Bit reset, write-one-to-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.

16.3.3.1.3 Outbound Descriptor Queue Dequeue Pointer Address Register (ODQDPAR)

The outbound descriptor queue dequeue pointer address register (ODQDPAR), shown in Figure 16-64, 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 (ODQDPAR[DQDPA] 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 pointers are equal after the dequeue pointer has been incremented, the queue is empty and the message unit halts until the enqueue pointer is incremented by software. 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.

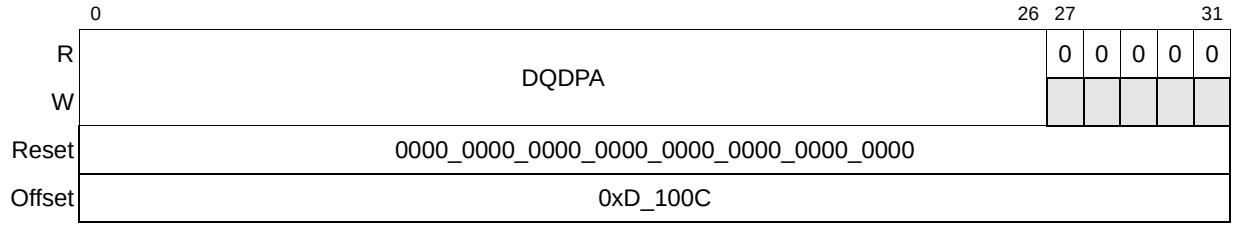


Figure 16-64. Outbound Descriptor Queue Dequeue Pointer Address Register (ODQDPAR)

Table 16-66 describes the fields of the ODQDPAR.

Table 16-66. 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

16.3.3.1.4 Outbound Unit Source Address Register (OSAR)

The outbound unit source address register (OSAR), shown in Figure 16-65, 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 address must be aligned to a double-word boundary, so the 3 lsbs are reserved.

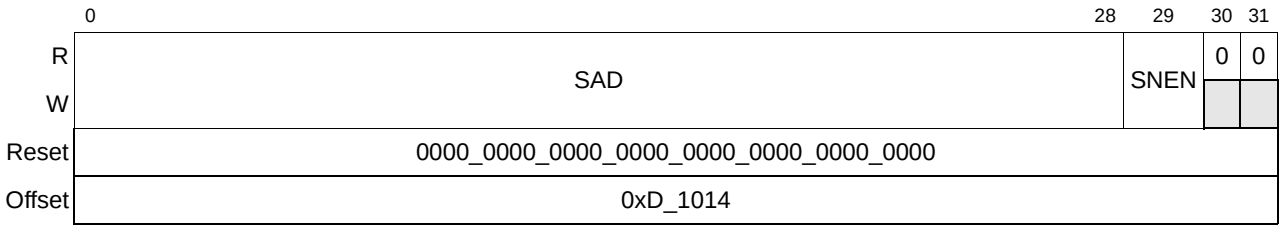


Figure 16-65. Outbound Unit Source Address Registers (OSAR)

Table 16-67 describes the fields of the OSAR.

Table 16-67. OSAR Field Descriptions

Bits	Name	Description
0–28	SAD	Source address. This is the source address of the message operation.
29	SNEN	Snoop enable 0 Snooping of the processor core disabled 1 Snooping of the processor core for data reads from local memory enabled
30–31	—	Reserved

16.3.3.1.5 Outbound Destination Port Register (ODPR)

The destination port register (ODPR), shown in Figure 16-66, indicates the port to which the message unit controller is to send data. Software must ensure that this is a valid port in the receiving device.

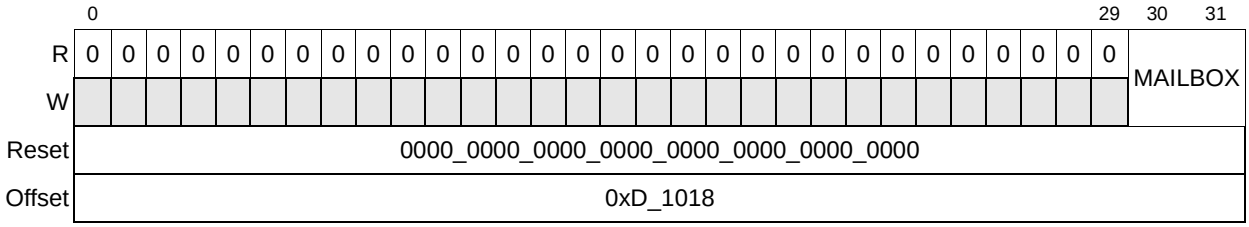


Figure 16-66. Outbound Destination Port Registers (ODPR)

Table 16-68 describes the fields of the ODPR.

Table 16-68. ODPR Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30–31	MAILBOX	Value for mbox field in message packet

16.3.3.1.6 Outbound Destination Attributes Register (ODATR)

The outbound destination attributes register (ODATR) contains the transaction attributes to be used for the message operation. Figure 16-67 shows the ODATR.

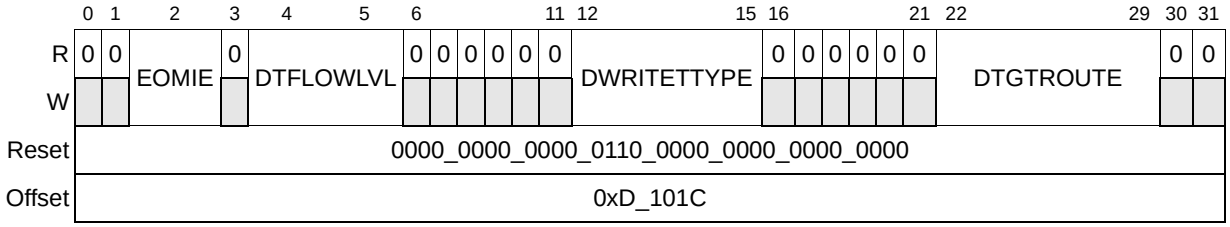


Figure 16-67. Outbound Destination Attributes Register (ODATR)

Table 16-69 describes the fields of the ODATR.

Table 16-69. 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.

Table 16-69. ODATR Field Descriptions (continued)

Bits	Name	Description
3	—	Reserved
4–5	DTFLOWLVL	Transaction flow priority 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

16.3.3.1.7 Outbound Double-Word Count Register (ODCR)

The outbound double-word count register (ODCR), shown in Figure 16-68, contains the number of double words for the message operation. The maximum message operation size is 4 Kbytes and the minimum is 8 bytes.

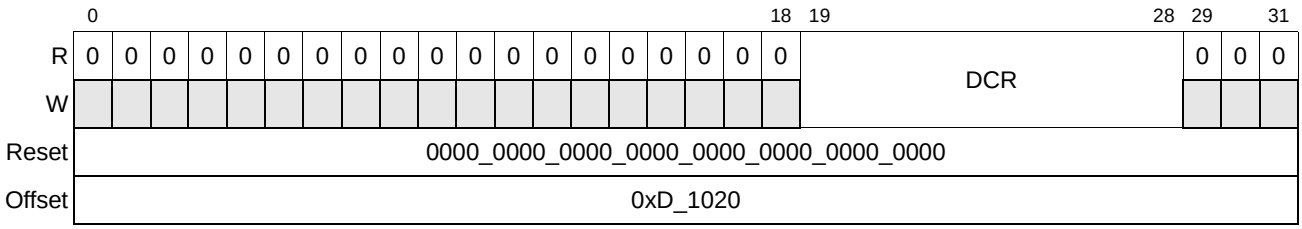


Figure 16-68. Outbound Double-Word Count Register (ODCR)

Table 16-70 describes the fields of the ODCR.

Table 16-70. 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 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

16.3.3.1.8 Outbound Descriptor Queue Enqueue Pointer Address Register (ODQEPAR)

The outbound descriptor queue enqueue pointer address register (ODQEPAR), shown in Figure 16-69, 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. If a message is ready to be sent, software writes a descriptor to the next location in the queue (indicated by the address in ODQEPAR), and then should set 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).

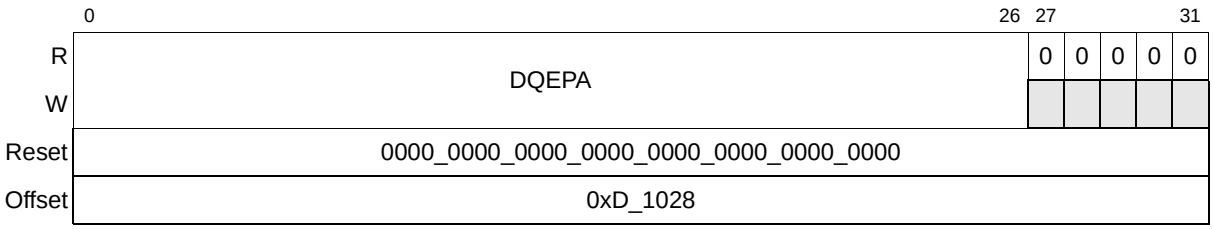


Figure 16-69. Outbound Descriptor Queue Enqueue Pointer Register (ODQEPAR)

Table 16-71 describes the fields of the ODQEPAR.

Table 16-71. 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

16.3.3.2 RapidIO Inbound Message Registers

Registers in this section describe the RapidIO inbound message registers.

16.3.3.2.1 Inbound Mode Register (IMR)

The inbound mode register (IMR) allows software to enable the mailbox controller and to control various message operation characteristics. Figure 16-70 shows the IMR.

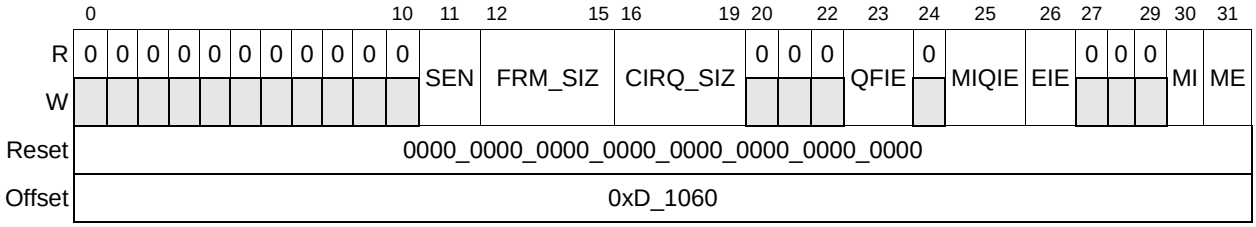


Figure 16-70. Inbound Mode Register (IMR)

Table 16-72 describes the fields of the IMR.

Table 16-72. IMR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	SEN	Snoop enable 0 Snooping of the processor core while writing messages into memory disabled 1 Snooping of the processor core 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. 0000–0001 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 1100–1111 Reserved
16–19	CIRQ_SIZ	Circular frame queue size. Determines the number of message frames that can be placed in 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 1011–1111 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)

Table 16-72. IMR Field Descriptions (continued)

Bits	Name	Description
26	EIE	Error interrupt enable 0 No interrupt is generated if a programming or transfer error is detected. 1 Generates an interrupt if a programming or transfer error is detected
27–29	—	Reserved
30	MI	Mailbox increment. Software sets MI after processing an inbound message. When software sets this bit, 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.

16.3.3.2.2 Inbound Status Register (ISR)

The inbound status register (ISR), shown in Figure 16-71, reports mailbox conditions during and after a message operation. Writing a 1 to a writable condition bit in ISR clears it.

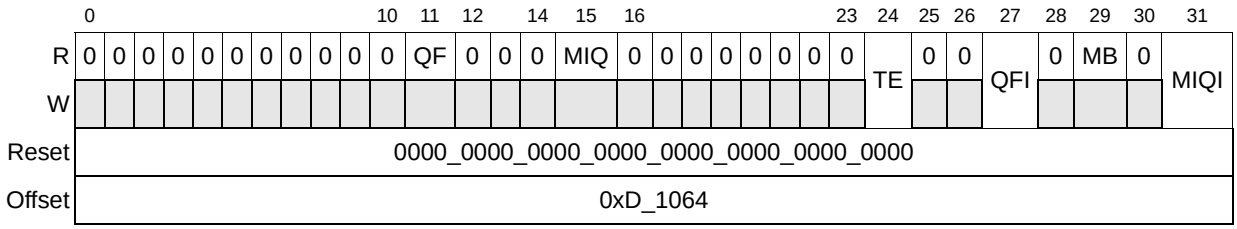


Figure 16-71. Inbound Status Register (ISR)

Table 16-73 describes the fields of the ISR.

Table 16-73. 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, this bit is set. Hardware clears this bit when software empties one or more entries.
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, this bit is set. Hardware clears this bit when software has emptied the queue.
16–23	—	Reserved

Table 16-73. ISR Field Descriptions (continued)

Bits	Name	Description
24	TE	Transaction error 0 No transaction error 1 Set when there is an error condition during the message operation. An interrupt is generated if IMR[IEI] is set. (Bit reset, write-one-to-clear)
25–26	—	Reserved
27	QFI	Queue full interrupt. (Bit reset, write-one-to-clear) 0 No queue full interrupt condition 1 If the queue becomes full and IMR[QFIE] is set, this bit is set and an interrupt is generated.
28	—	Reserved
29	MB	Mailbox busy (read only) 0 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. (Bit reset, write-one-to-clear) 0 No message-in-queue interrupt condition 1 If the queue goes from empty to not empty and IMR[MIQIE] is set, this bit is set and an interrupt is generated.

16.3.3.2.3 Inbound Frame Queue Dequeue Pointer Address Register (IFQDPAR)

The inbound frame queue dequeue pointer address register (IFQDPAR), shown in Figure 16-72, 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, software sets IMR[MI]. The mailbox controller then increments IFQDPAR to point to the next frame in memory and clears MI. If the inbound frame queue enqueue pointer and the inbound frame queue dequeue pointer are not equal (indicating that the queue is not empty), software can read the next message frame from memory for processing. If the enqueue and dequeue pointers are equal after being incremented by software, the queue is empty and all outstanding messages have been processed.

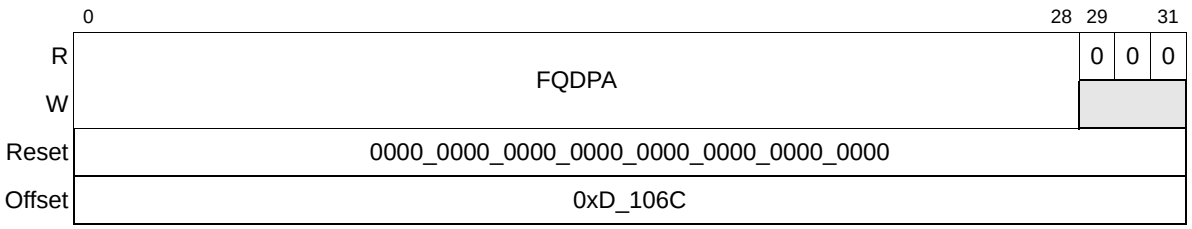


Figure 16-72. Inbound Frame Queue Dequeue Pointer Address Registers (IFQDPAR)

Table 16-74 describes the fields of the IFQDPAR.

Table 16-74. 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

16.3.3.2.4 Inbound Frame Queue Enqueue Pointer Address Register (IFQEPAR)

The inbound frame queue enqueue pointer address register (IFQEPAR), shown in Figure 16-73, 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 the mailbox controller receives a message, 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 the following:

- If the enqueue and dequeue pointers match, the queue is now full and the mailbox controller does not accept 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.

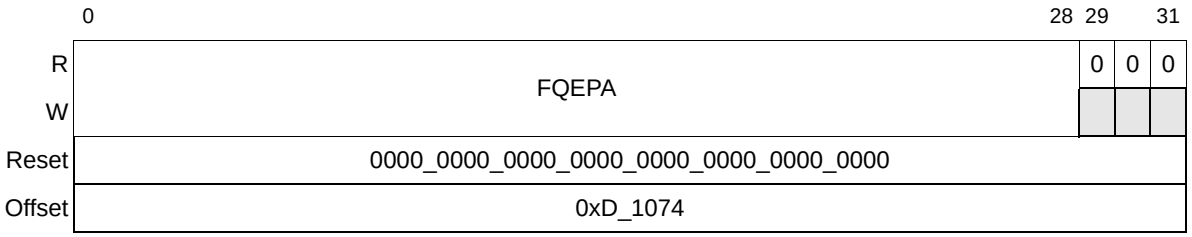


Figure 16-73. Inbound Frame Queue Enqueue Pointer Address Registers (IFQEPAR)

Table 16-75 describes the fields of the IFQEPAR.

Table 16-75. 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

16.3.3.3 RapidIO Doorbell Registers

These registers control the RapidIO doorbell unit. The following sections provide descriptions of these registers.

16.3.3.3.1 Doorbell Mode Register (DMR)

The doorbell mode register (DMR) allows software to enable the doorbell controller to control various doorbell operation characteristics. Figure 16-74 shows the DMR.

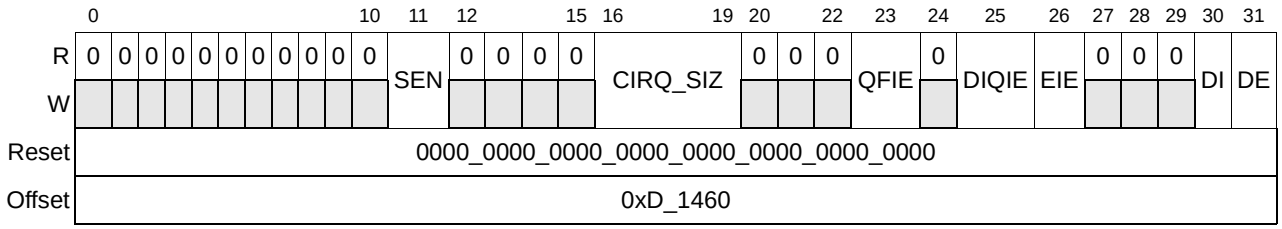


Figure 16-74. Doorbell Mode Register (DMR)

Table 16-76 describes the fields of the DMR.

Table 16-76. DMR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	SEN	Snoop enable 0 Snooping of the processor core while writing doorbell information into memory disabled 1 Snooping of the processor core 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 in 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 1011–1111 Reserved
20–22	—	Reserved

Table 16-76. DMR Field Descriptions (continued)

Bits	Name	Description
23	QFIE	Queue full interrupt enable 0 No QFI interrupt 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
25	DIQIE	Doorbell in queue interrupt enable 0 No DIQ interrupt 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	EIE	Error interrupt enable 0 No interrupt is generated if a programming or transfer error is detected. 1 Generates an interrupt if a programming or transfer error is detected
27–29	—	Reserved
30	DI	Doorbell increment. Software sets DI after processing an inbound doorbell. When software sets DI, hardware increments the DQDPAR and clears DI. 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.

16.3.3.3.2 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. Figure 16-75 shows the DSR.

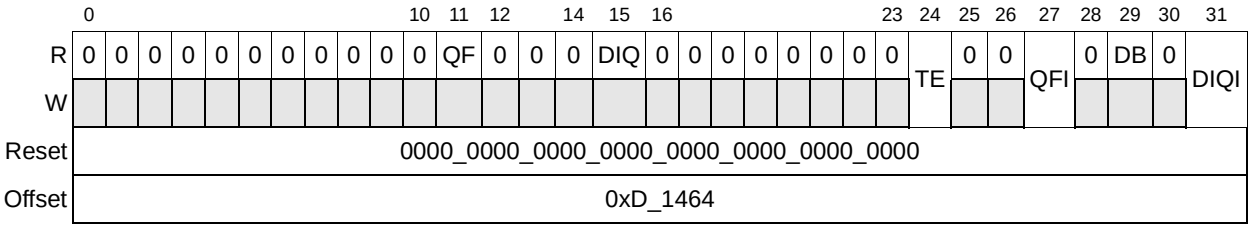


Figure 16-75. Doorbell Status Register (DSR)

Table 16-77 describes the fields of the DSR.

Table 16-77. DSR 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, this bit is set. Hardware clears this bit when software empties one or more entry.
12–14	—	Reserved

Table 16-77. DSR Field Descriptions (continued)

Bits	Name	Description
15	DIQ	Doorbell-in-queue (read only) 0 No doorbell-in-queue conditions 1 If the queue goes from empty to not empty, this bit is set. Hardware clears this bit when software has emptied the queue.
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. An interrupt is generated if DMR[IEI] is set.(Bit reset, write-one-to-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-one-to-clear).
28	—	Reserved
29	DB	Doorbell busy (read only) 0 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-one-to-clear)

16.3.3.3.3 Doorbell Queue/Dequeue Pointer Address Register (DQDPAR)

The doorbell queue dequeue pointer address register (DQDPAR), shown in Figure 16-76, 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, software sets DMR[DI]. The doorbell hardware then increments the DQDPAR 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), software can read the next doorbell from memory for processing. If the enqueue and dequeue pointers are equal after being incremented by software, the queue is empty and all outstanding doorbells have been processed.

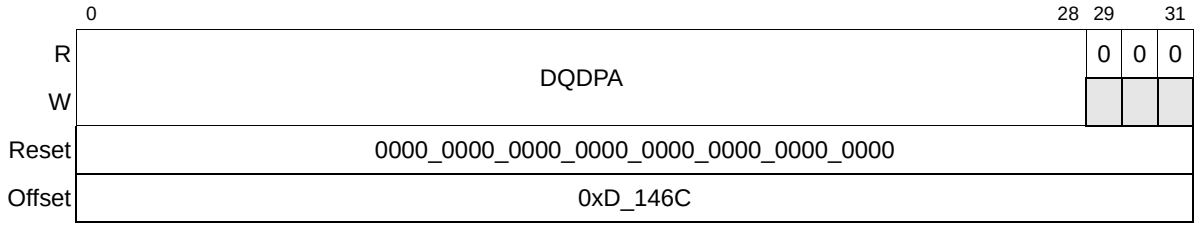


Figure 16-76. Doorbell Queue Dequeue Pointer Address Registers (DQDPAR)

Table 16-78 describes the fields of the DQDPAR.

Table 16-78. 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

16.3.3.3.4 Doorbell Queue Enqueue Pointer Address Register (DQEPAR)

The doorbell queue enqueue pointer address register (DQEPAR), shown in Figure 16-77, 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 the doorbell controller receives a packet, 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. This can result in the following:

- 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, DSR[MIQI] is set and an interrupt is generated.

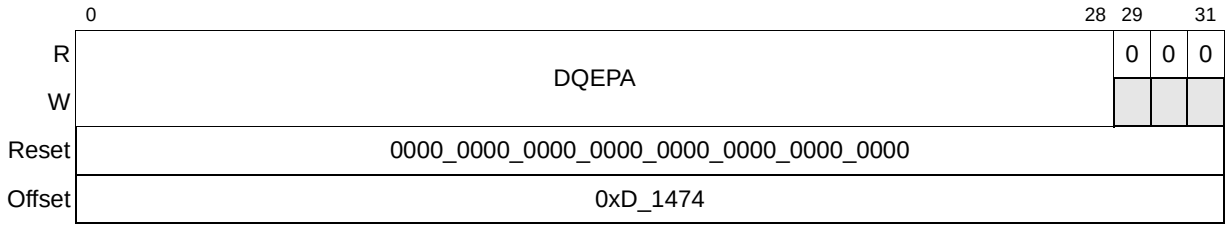


Figure 16-77. Doorbell Queue Enqueue Pointer Address Register (DQEPAR)

Table 16-79 describes the fields of the DQEPAR.

Table 16-79. 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

16.3.3.4 RapidIO Port-Write Registers

These registers control the RapidIO port-write unit. The following sections describe these registers. Additional information may be found in the *Parallel RapidIO Intconnect Specification*.

16.3.3.4.1 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. Figure 16-78 shows the PWMR.

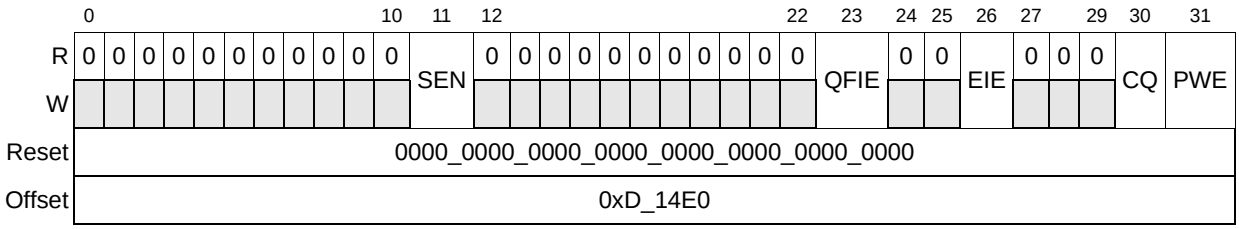


Figure 16-78. Port-Write Mode Register (PWMR)

Table 16-80 describes the fields of the PWMR.

Table 16-80. PWMR Field Descriptions

Bits	Name	Description
0–10	—	Reserved
11	SEN	Snoop enable 0 Snooping the processor core while writing port-write data payload into memory disabled 1 Snooping the processor core while writing port-write data payload into memory enabled
12–22	—	Reserved
23	QFIE	Queue full interrupt enable 0 No QFI interrupt generated 1 Generates an interrupt when the queue becomes full (that is, the controller has written a port-write data payload into memory)
24–25	—	Reserved
26	EIE	Error interrupt enable 0 No interrupt is generated if a programming or transfer error is detected. 1 Generates an interrupt if a programming or transfer error is detected

Table 16-80. PWMR Field Descriptions (continued)

Bits	Name	Description
27–29	—	Reserved
30	CQ	Clear queue. Set by software after processing an inbound port-write operation. Hardware clears the previous port-write operation from the port-write queue and clears CC. Always reads as 0 if PWE = 1.
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.

16.3.3.4.2 Port-Write Status Register (PWSR)

The port-write status register (PWSR), shown in Figure 16-79, reports various port-write conditions after a port-write operation. Writing a 1 to a corresponding set bit clears the bit.

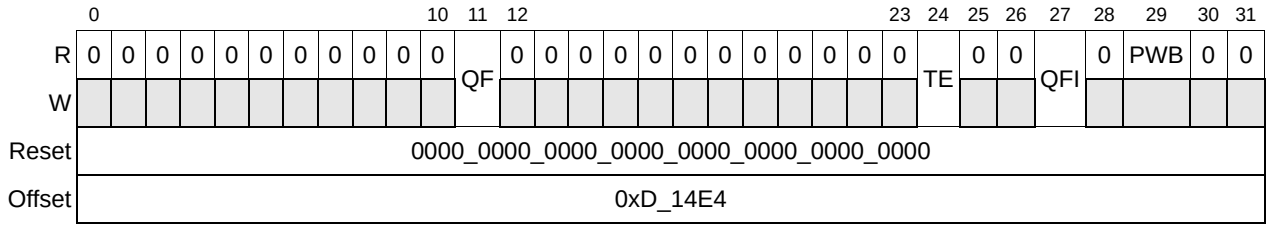


Figure 16-79. Port-Write Status Register (PWSR)

Table 16-81 describes the fields of the PWSR.

Table 16-81. 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 0 No transaction error occurred 1 There was an error condition (the port-write transaction is greater than 64 bytes) during the port-write operation. An interrupt is generated if PWMR[IEI] is set. (Bit reset, write-one-to-clear)
25–26	—	Reserved
27	QFI	Queue full interrupt. (Bit reset, write-one-to-clear) 0 Must be cleared by software after a port-write is serviced to re-enable the port-write controller 1 The queue became full and PWMR[QFIE] is set. An interrupt is generated.
28	—	Reserved

Table 16-81. PWSR Field Descriptions (continued)

Bits	Name	Description
29	PWB	Port-write busy (read only) 0 Cleared as a result of receiving an error or upon the completion of a port-write operation 1 Indicates that a port-write operation is currently in progress
30–31	—	Reserved

16.3.3.4.3 Port-Write Queue Base Address Register (PWQBAR)

The port-write queue base address register (PWQBAR), shown in Figure 16-80, contains the 64-byte cache line address for the port-write data payload. PWQBAR must be initialized by software.

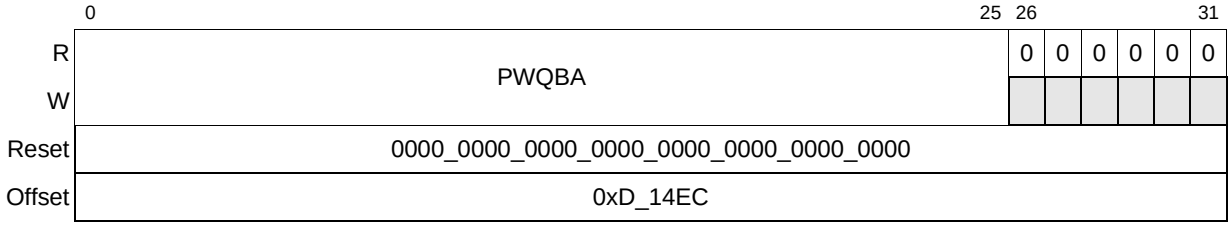


Figure 16-80. Port-Write Queue Base Address Register (PWQBAR)

Table 16-82 describes the fields of the PWQBAR.

Table 16-82. 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

The following sections provide summaries of RapidIO transactions, packets and control symbols. Additional information may be found in the *Parallel RapidIO Intconnect Specification.*

16.4.1 RapidIO Transaction, Packet, and Control Symbol Summary

Table 16-83 summarizes the RapidIO transaction types supported by the RapidIO controller. The table lists the packet format type, the RapidIO transaction, the transaction programming model, 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 Table 16-83.

Table 16-83. RapidIO Transaction Summary

Packet Format	RapidIO Transaction	Programming Model	Possible Processor Operations	Possible RapidIO Responses	Comments
2 Non-intervention request class	I/O 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 I/O 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 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
	atomic swap	IO	N/A	N/A	The RapidIO controller does not 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
8 Maintenance class	maintenance read	All	M = 0 read type operation	done, error	Although possible, maintenance reads from memory locations outside of the RapidIO architectural memory space (0xC_0000–0xC_FFFC) return unreliable data.
	maintenance write	All	M = 0 write type operation	done, error	None
10 Doorbell class	doorbell	Message passing	M = 0 write type operation	done, retry, error	None

Table 16-83. RapidIO Transaction Summary (continued)

Packet Format	RapidIO Transaction	Programming Model	Possible Processor Operations	Possible RapidIO Responses	Comments
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 16-84 summarizes packet formats. The following definitions and Figure 16-81 accompany the table.

- PMF—Physical maintenance field; prefixed to each packet format (see Figure 16-81)
- TYPE —Packet type (1, 2, 5, 6, 8, 10, 11, 13)
- TARID —target_id: the receiver's ID within its domain
- SRCTID— 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
- 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 8 bits: letter (2) + mbox (2) + msgseg (4)
- W—Word pointer; used in the decoding of the transaction size

S = 0	AckID	0100	prio	TT = 00
1	3	4	2	2

Figure 16-81. Physical Maintenance Field (PMF) Definition

Table 16-84. RapidIO Packet Format Summary (TT = 0b00)

<table>
<tr><th>Type</th><th>Packet Format</th></tr>
<tr><td>2
(0010)</td><td>Request To Home:
<table>
<tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td>TTYPE</td><td>RDSIZE</td><td>SRCTID</td></tr>
<tr><td>12</td><td>4</td><td>8</td><td>8</td><td>4</td><td>4</td><td>8</td></tr>
</table>
<table>
<tr><td>ADDRESS</td><td>W</td><td>ADDRESS</td></tr>
<tr><td>29</td><td>1</td><td>2</td></tr>
</table>
TTYPE:0010 (I/O read home); 0100 (nread); 1100 (atomic_inc); 1101 (atomic_dec); 1110 (atomic_set); 1111 (atomic_clr)</td></tr>
<tr><td>5
(0101)</td><td>Write Class:
<table>
<tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td>TTYPE</td><td>WRSIZE</td><td>SRCTID</td></tr>
<tr><td>12</td><td>4</td><td>8</td><td>8</td><td>4</td><td>4</td><td>8</td></tr>
</table>
<table>
<tr><td>ADDRESS</td><td>W</td><td>ADDRESS</td><td>MSB for DW0 --> LSB for DWn</td></tr>
<tr><td>29</td><td>1</td><td>2</td><td></td></tr>
</table>
TTYPE:0001 (flush); 0010–0011 (Reserved); 0100 (nwrite); 0101 (nwrite_r); 0110–0111 (Reserved);</td></tr>
<tr><td>6
(0110)</td><td>Streaming-Write Class:
<table>
<tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td>ADDRESS</td></tr>
<tr><td>12</td><td>4</td><td>8</td><td>8</td><td>29</td></tr>
</table>
<table>
<tr><td></td><td>ADDRESS</td><td>MSB for DW0 --> LSB for DWn</td></tr>
<tr><td>1</td><td>2</td><td></td></tr>
</table></td></tr>
<tr><td>8
(1000)</td><td>Maintenance Class Request:
<table>
<tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td>TTYPE</td><td>RDSIZE</td><td>SRCTID</td></tr>
<tr><td>12</td><td>4</td><td>8</td><td>8</td><td>4</td><td>4</td><td>8</td></tr>
</table>
<table>
<tr><td>HOPCNT</td><td>CONFIG_OFFSET</td><td>W</td><td></td><td>MSB for DW0 --> LSB for DWn</td></tr>
<tr><td>8</td><td>21</td><td>1</td><td>2</td><td></td></tr>
</table>
TTYPE: 0000 (read); 0001 (write): 0100 (port-write)

Maintenance Class Response:
<table>
<tr><td>PMF</td><td>TYPE</td><td>TARID</td><td>SRCID</td><td>TTYPE</td><td>STATUS</td></tr>
<tr><td>12</td><td>4</td><td>8</td><td>8</td><td>4</td><td>4</td></tr>
</table>
<table>
<tr><td>TARTID</td><td>HOPCNT</td><td></td><td>MSB for DW0 --> LSB for DWn</td></tr>
<tr><td>8</td><td>8</td><td>24</td><td></td></tr>
</table>
TTYPE: 0010 (read response); 0011 (write response)
STATUS: 0000 (done); 0111 (error);</td></tr>
</table>

Table 16-84. RapidIO Packet Format Summary (TT = 0b00) (continued)

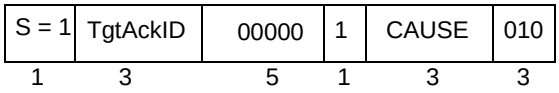
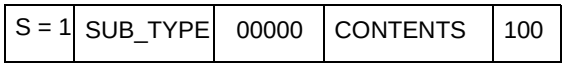
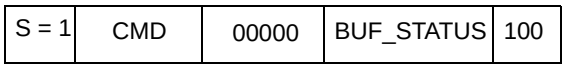
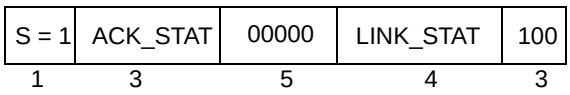
Type	Packet Format
10 (1010)	Doorbell Class: PMF (12) | TYPE (4) | TARID (8) | SRCID (8) | (8) | SRCTID (8) INFO (16)
11 (1011)	Message Class: PMF (12) | TYPE (4) | TARID (8) | SRCID (8) | MSGLEN (4) | SSIZE (4) | LETTER (2) MBOX (2) | MSGSEG (4) | MSB for DW0 --> LSB for DWn
13 (1101)	Response Class: PMF (12) | TYPE (4) | TARID (8) | SRCID (8) | TTYPE (4) | STATUS (4) TARTID (8) | MSB for DW0 --> LSB for DWn 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 16-85 summarizes the defined control symbols. Implementation details for this particular RapidIO controller are noted in the comments section of the table.

Table 16-85. Control Symbol Formats

Symbol Type	Symbol Format	Comments
0 (000)	Packet accepted: S = 1 (1) | TgtAckID (3) | 00000 (5) | BUF_STATUS (4) | 000 (3)	The RapidIO controller sets buf_status to 0xF when it generates a packet accepted control symbol.
1 (001)	Packet retry: S = 1 (1) | TgtAckID (3) | 00000 (5) | 0000 (4) | 001 (3)	None

Table 16-85. Control Symbol Formats (continued)

Symbol Type	Symbol Format	Comments
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 (The RapidIO controller sets to 1111) 001 Stomp (unused) 010 EOP (The RapidIO controller 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:  	The RapidIO controller sets buf_status to 0xF when it generates a link request command. 000 Send training 010 Override width 011 Reset 100 Input status Note that the RapidIO controller causes a checkstop if a software-initiated send training link request command collides with an inbound send training link request which is immediately followed by an inbound training pattern.
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 Valid, ackID 0 .... 111 Valid, ackID 7

16.5 RapidIO Functionality

This section describes RapidIO general device functionality, common transport layer functionality, and logical layer functionality.

16.5.1 General Functionality Lists

The general device functionality lists are applicable to devices of all classes, including switch devices without end points. The following tables list the functions that the RapidIO controller supports.

Table 16-86 lists general requirements for a RapidIO-compliant device.

Table 16-86. General Device Functionality List

Item No.	Item Functionality	Meets Reqs?
1	All device generated packets adhere to the bit streams defined in the appropriate logical, transport, and physical layer specifications.	Y
2	All device-generated control symbols 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

Table 16-86. General Device Functionality List (continued)

Item No.	Item Functionality	Meets Reqs?
20	Writes to implementation-defined CSR and extended features register bits are determined by the implementation.	Y
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 are implementation-defined.	Y

16.5.1.1 8/16 LP-LVDS Layer Functionality Lists

The physical layer 8/16 LP-LVDS (low-power, low-voltage differential signaling) layer functionality is subdivided into individual lists.

Table 16-87 lists requirements for basic 8/16 LP-LVDS physical layer functionality for the device. Note that for device ID-related functions, the device ID for the RapidIO controller is set by power-on reset signals as described in Section 4.4.3.11, “RapidIO Device ID.”

Table 16-87. 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 follows the link initialization and training procedure	Y
	7A ACTION: Device responds 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 can 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 16-87. 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 cannot 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-dependent (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 dependent	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 dependent	Y
15	Device follows system initialization and exploration inter-operability procedure	Y
	15A. ACTION: End point agent device that contains a boot ROM is initially assigned base device ID = 0xFE at power-on reset	Y
	15B. ACTION: End point agent device without a boot ROM is initially assigned base device ID = 0xFF at power-on reset	Y
	15C. ACTION: System host device is initially assigned a non-reserved base device ID at power-on reset (end point device only)	Y
	15D. ACTION: Host device sets PGCCSR[D] at power-on reset (end point device only)	Y
	15E. ACTION: Agent device resets PGCCSR[D] at power-on reset (end point device only)	Y
	15F. ACTION: Host device sets the master enable bit (PGCCSR[M]) at power-on reset (end point device only)	Y
	15G. ACTION: Agent device resets the master enable bit (PGCCSR[M]) at power-on reset (end point device only)	Y
	15H. ACTION: End point device responds 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 sends maintenance responses that it generates to the port that the maintenance request was received on	Y

Table 16-88 lists requirements for link maintenance functionality and the compliance of the RapidIO controller.

Table 16-88. 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 does not issue more than one outstanding link-request control symbol (except in case of time-out)	Y
	6A. DETAIL: Device holds off another link-request control symbol until expected link-response control symbol is received	Y

Table 16-89 lists requirements for the device while transmitting a packet (a general device is required to generate response packets) and the compliance of the RapidIO controller.

Table 16-89. General Device 8/16 LP-LVDS Physical Layer Packet Transmission List

Item No.	Item Functionality	Meets Reqs?
1	Device cannot issue more than 8 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 3 reserved bits are cleared.	Y

Table 16-89. General Device 8/16 LP-LVDS Physical Layer Packet Transmission List (continued)

Item No.	Item Functionality	Meets Reqs?
5	A response packet for a request packet never 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: Moves from the port OK state to the output retry-stopped state when a packet-retry control symbol is received	Y
	13B. ACTION: Completes transmission of current packet if applicable; don't transmit new packets	Y
	13C. ACTION: Sends a restart-from-retry control symbol to the receiving device	Y
	13D. ACTION: Moves from the output retry-stopped state to the port OK state upon completion of the output port internal retry recovery procedure	Y
	13E. ACTION: Begins retransmitting packets from the returned expected ackID value	Y
	13F. DETAIL: Retried packet is eventually re-transmitted (end point devices only; not true for cut-through switch devices)	Y

Table 16-90 lists the requirements for the device to receive a packet.

Table 16-90. 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	Starts with ackID = 0 after reset	Y
3	Every packet causes an acknowledge control symbol with the corresponding ackID field.	Y
4	Device sends acknowledge control symbols in same order as packets are received (ackIDs are issued sequentially)	Y
5	Device does 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 3 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

Table 16-90. General Device 8/16 LP-LVDS Physical Layer Packet Reception List (continued)

Item No.	Item Functionality	Meets Reqs?
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: do not 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: Moves from the port OK state to the input retry-stopped state upon detection of a packet retry situation	Y
	12C. ACTION: Sends 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 detects control symbol errors while in input retry-stopped state	Y
	12F. ACTION: Moves 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 is not 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

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 16-91 lists recoverable errors at the physical layer.

Table 16-91. 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: Moves from the port OK state to the input error-stopped state	Y
	1B. ACTION: Sends 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: Moves from the input error-stopped state to the port OK state	Y
2	Device encounters S-bit parity failure	Y
	2A. ACTION: Moves from the port OK state to the input error-stopped state	Y
	2B. ACTION: Sends 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: Moves from the input error-stopped state to the port OK state	Y
3	Device encounters incorrect CRC for a packet	Y
	3A. ACTION: Moves from the port OK state to the input error-stopped state	Y
	3B. ACTION: Sends 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: Moves from the input error-stopped state to the port OK state	Y
4	Device encounters unexpected ackID value in a packet	Y
	4A. ACTION: Moves from the port OK state to the input error-stopped state	Y
	4B. ACTION: Sends 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: Moves from the input error-stopped state to the port OK state	Y
5	Device encounters a packet that exceeds the 256 byte maximum packet size limit	Y
	5A. ACTION: Moves from the port OK state to the input error-stopped state	Y
	5B. ACTION: Sends 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: Moves from the input error-stopped state to the port OK state	Y

Table 16-91. General Device 8/16 LP-LVDS Physical Layer Recoverable Errors List (continued)

Item No.	Item Functionality	Meets Reqs?
6	Device encounters a corrupt control symbol	Y
	6A. ACTION: Moves from the port OK state to the input error-stopped state	Y
	6B. ACTION: Sends 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: Moves 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: Moves 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: Moves from the output error-stopped state to the port OK state	Y
	8E. ACTION: Starts re-transmitting at the ackID value returned with the received link-response control symbol	Y
9	Device encounters a packet-not-accepted control symbol	Y
	9A. ACTION: Moves 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: Moves from the output error-stopped state to the port OK state	Y
	9E. ACTION: Starts re-transmitting at the ackID value returned with the received link-response control symbol	Y

Table 16-91. General Device 8/16 LP-LVDS Physical Layer Recoverable Errors List (continued)

Item No.	Item Functionality	Meets Reqs?
10	Device encounters an acknowledge control symbol time-out	Y
	10A. ACTION: Moves 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: Moves from the output error-stopped state to the port OK state	Y
	10E. ACTION: Starts re-transmitting at the ackID value returned with the received link-response control symbol	Y
11	Device encounters any transmission error during error recovery	Y

Relatively few nonrecoverable errors occur at the physical layer. Behavior of the interface is not defined for these events. Table 16-92 lists nonrecoverable errors at the physical layer.

Table 16-92. 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	Time-out between link-request control symbol and link-response control symbol	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 subdivided into individual lists.

Table 16-93 lists the basic functionality for the device at the common transport layer and the compliance of the RapidIO controller.

Table 16-93. 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 dependent (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

Table 16-94 lists the functionality for this controller in transmitting a packet (a slave device is required to generate response packets).

Table 16-94. 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

Table 16-95 lists the functionality for this controller in receiving a packet.

Table 16-95. 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

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 16-96 lists detectable errors.

Table 16-96. 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 subdivided into individual lists.

Table 16-97 lists the requirements for basic device functionality and the compliance of this controller. A class 1 device is always the target of a request transaction.

Table 16-97. 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
9	All necessary CARs (capabilities registers) exist and can be read.	Y
	9A. DETAIL: Device identity CAR; value is vendor- and implementation-dependent.	Y
	9B. DETAIL: Device information CAR; value is vendor- and implementation-dependent	Y
	9C. DETAIL: Assembly identity CAR; value is vendor- and implementation-dependent	Y
	9D. DETAIL: Assembly information CAR; value is vendor- and implementation-dependent	Y
	9E. DETAIL: Processing element features CAR; value is implementation-dependent; must indicate support for 34-bit address format packets	Y
	9F. DETAIL: Switch port information CAR; value is implementation-dependent (switch devices only)	Y
	9G. DETAIL: Source operations CAR; value is implementation-dependent (end point devices only)	Y
	9H. DETAIL: Destination operations CAR; value is implementation-dependent (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 zeros).	Y
13	All extended features blocks lie 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 zeros	Y

Table 16-98 lists requirements for this controller when functioning as a transaction target.

Table 16-98. 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 4 bytes	Y
	1B. DETAIL: Maintenance read request generates a maintenance read response	Y
	1C. DETAIL: Maintenance read response data payload is 4 bytes	Y
2	Maintenance write transaction	Y
	2A. DETAIL: Maintenance write request may be for 4 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

16.5.3.1 Logical Layer Source Transaction Support List

General functionality does not require a device to initiate any operations.

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 16-99 lists the errors detected by the RapidIO controller.

Table 16-99. 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

16.5.3.2 Logical Layer Extended Functionality

The RapidIO controller supports I/O read and flush with data GSM transactions, and all I/O and message transactions as described by the *Parallel RapidIO Intconnect Specification* except for atomic test and swap and atomic swap transactions.

16.6 RapidIO Errors

RapidIO errors are classified under three categories:

- Recoverable errors—The RapidIO controller supports hardware error detection and recovery mechanism as specified in the *Parallel RapidIO Intconnect Specification*. The RapidIO controller 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. See Table 16-100 for specifics regarding the RapidIO controller.
- Notification errors—The RapidIO controller detects many hardware nonrecoverable errors that are non-fatal such as exceeded threshold counts (as defined in PERTR and PRTR). Upon detection of a notification error the appropriate bit is set in the port notification/fatal error detect register (PNFEDR) and an interrupt is generated if enabled in the port notification/fatal error interrupt enable register (PNFEIER). Hardware continues to operate after notification errors.
- Fatal errors—The RapidIO controller 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 (PECSR[V] indicates successful capture) in the error packet capture registers (EPCR*n*) if the appropriate error detect is enabled (PNFEDiR). An interrupt is generated if the appropriate fatal error interrupt is enabled in the port notification/fatal error interrupt enable register (PNFEIER). The RapidIO hardware either goes into training mode, or remains in the OK state based on the error type.

If the valid bit of the port error capture status register (PECSR[V]) is set, the information in error packet capture registers EPCR0–EPCR3 is valid:

Fatal errors caused by a bad link invoke training in an attempt to recover the link. If communication is not re-established, a RESET is required to reconcile RapidIO's internal queues to resume normal operation. Table 16-100 lists the recoverable errors detected by the RapidIO implementation and the resulting actions taken.

Table 16-100. Recoverable Errors Detected by RapidIO Controller

Error	Description	Action	Capture Registers
Inbound corrupt control symbol	Received a control symbol in which true does not match complement counterpart	Sends "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	Sends "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	Starts link recovery process	No information logged in capture register(s)

Table 16-100. Recoverable Errors Detected by RapidIO Controller (continued)

Error	Description	Action	Capture Registers
Inbound packet with unexpected ackID	Received packet with unexpected ackID value (out-of-sequence ackID)	Sends “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	Sends “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	Sends “PACKET_NOT_ ACCEPTED” ack with “General error” cause	No information logged in capture register(s)
Inbound packet exceeds 256 bytes	Received packet that exceeds the maximum allowed size by the *Parallel RapidIO Intconnect Specification*. This controller performs this function by detecting a data payload greater than 256 bytes.	Sends “PACKET_NOT_ ACCEPTED” ack with “General error” cause	No information logged in capture register(s)
Packet-not-accepted: Encountered internal error	Received ack control symbol with packet-not-accepted of type “encountered internal error”	Starts 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”	Starts 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”	Starts 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”	Starts 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”	Starts 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”	Starts 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.	Starts 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	Starts 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	Starts link recovery process	No information logged in capture register(s)

Table 16-100. Recoverable Errors Detected by RapidIO Controller (continued)

Error	Description	Action	Capture Registers
Ack before restart-from-retry	Received an ack control symbol after receiving an ack retry and before sending a restart-from-retry	Starts link recovery process	No information logged in capture register(s)
Unexpected training symbol	Received a training symbol while in the “OK state”	Starts link recovery process	No information logged in capture register(s)
Unexpected stomp symbol	Received a stomp control symbol while there is no packet being received	Starts 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.	Starts 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	Starts 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.	Starts link recovery process	No information logged in capture register(s)
Frame toggle alignment	Received FRAME signal toggles at a non 32-bit boundary	Sends “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	Sends “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.	Starts link recovery process	No information logged in capture register(s)

Table 16-101 lists the notification errors detected by the RapidIO implementation and the resulting actions taken.

Table 16-101. Notification Errors Detected by RapidIO Controller

Error	Description	Action	Capture Registers
Error recovery threshold error	Error recovery threshold count (defined in PERTR[RCTT]) exceeded	Generates interrupt if enabled (PNFEIER[ETIE] is cleared)	No information logged in capture register(s)
Retry threshold error	Consecutive ack retry threshold count (defined in PRTR[RTT]) exceeded	Generates interrupt if enabled (PNFEIER[RTIE] is cleared)	No information logged in capture register(s)
Received request ERROR response	Received a response of type error for anything other than a message	Generates interrupt if enabled (PNFEIER[RERIE] is cleared)	Information logged in SYSINT capture registers.

Table 16-102 lists the fatal errors detected by the RapidIO implementation and the resulting actions taken.

Table 16-102. Fatal Errors Detected by RapidIO Controller

Error	Description	Action	Capture Registers
Link response time-out	A link response is not received within the specified time-out interval.	Generates interrupt if enabled (PNFEIER[LTIE] is cleared)	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.	Generates interrupt if enabled (PNFEIER[RSTIE] is cleared)	No information logged in capture registers.
Missing idle after training	Idle not received after a requested training sequence completes	Generates interrupt if enabled (PNFEIER[NTIIIE] is cleared)	No Information logged in capture register(s)
Message segment error	Received a message packet with the segment field greater than the message length field (msgseg > msglen)	Generates interrupt if enabled (PNFEIER[SIE] is cleared)	No information logged in capture register(s)
Duplicate message segment	Received a duplicate message segment from RapidIO	Generates interrupt if enabled (PNFEIER[DSIE] is cleared)	No information logged in capture register(s)
Message length error	Received a message packet from RapidIO with a bad message length	Generates interrupt if enabled (PNFEIER[BMIE] is cleared)	No information logged in capture register(s)
Nonsensical ackID	AckID received with link response does not make sense	Generates interrupt if enabled (PNFEIER[NAIE] is cleared)	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. Examples include: illegal transaction types, illegal sizes	Generates interrupt if enabled (PNFEIER[IRFIE] is cleared)	Information logged in RapidIO capture register(s)
Unexpected request data	Read type with a data payload	Generates interrupt if enabled (PNFEIER[URDIE] is cleared)	Information logged in RapidIO capture register(s)
Unsupported request	Unsupported packet format type, or transport type (TT bits)	Generates interrupt if enabled (PNFEIER[URIE] is cleared)	Information logged in RapidIO capture register(s)
Unexpected request size	Packet data is not expected size	Generates interrupt if enabled (PNFEIER[URSIE] is cleared)	Information logged in RapidIO capture register(s)
Illegal write request	Write types with no data payload.	Generates interrupt if enabled (PNFEIER[IWRIE] is cleared)	Information logged in RapidIO capture register(s)
Illegal response fields	Illegal combinations of fields in the response packet	Generates interrupt if enabled (PNFEIER[IRSFIE] is cleared)	Information logged in RapidIO capture register(s)
Illegal response type	Illegal response for a given request type	Generates interrupt if enabled (PNFEIER[IRTIE] is cleared)	Information logged in RapidIO capture register(s)
Bad read response	Read “done” response with no data payload	Generates interrupt if enabled (PNFEIER[BRRIE] is cleared)	Information logged in RapidIO capture register(s)
Bad write response	Write response with a data payload	Generates interrupt if enabled (PNFEIER[BWRIE] is cleared)	Information logged in RapidIO capture register(s)

Table 16-102. Fatal Errors Detected by RapidIO Controller (continued)

Error	Description	Action	Capture Registers
Unexpected response data size	Response packet data is not expected size	Generates interrupt if enabled (PNFEIER[URSSIE] is cleared)	Information logged in RapidIO capture register(s)
Unsolicited response without data	A response without data is received while there is no corresponding request that matches that TID	Generates interrupt if enabled (PNFEIER[URESIE] is cleared)	Information logged in RapidIO capture register(s)
Unsolicited response with data	A response with data is received while there is no corresponding request that matches that TID	Generates interrupt if enabled (PNFEIER[URSDIE] is cleared)	Information logged in RapidIO capture register(s)
Message error response	Received a response of type error for an outbound message packet	Generates interrupt if enabled (PNFEIER[MERIE] is cleared)	Information logged in RapidIO capture register(s)
Message descriptor fetch error	A message descriptor fetch from local memory gets an error (for example, ECC).	Generates interrupt if enabled (PNFEIER[MDFIE] is cleared)	No information logged in RapidIO capture register(s)
Message size error	Received message packet data payload is not of the size specified in the ssize field (with the exception of the last packet which may be less)	Generates interrupt if enabled (PNFEIER[MSIE] is cleared)	Information logged in RapidIO capture register(s)
Illegal transaction target error	Received a packet whose target ID does not match RapidIO deviceID while accept_all mode is disabled	Generates interrupt if enabled (PNFEIER[ITIE] is cleared)	Information logged in RapidIO capture register(s)

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 *Parallel RapidIO Intconnect Specification* allows for 48- and 64-bit addresses, this 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 is 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 4 Kbytes and the largest is 4 Gbytes.

16.7.1 Outbound ATMU Translation

The nine outbound ATMU windows (0–8) perform the mapping from the internal 32-bit address space to the 34-bit address space of RapidIO. They also map attributes such as transaction type and priority level. 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

Note that some 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.

16.7.1.2 Outbound Special Transactions and Requirements

Due to requirements of the *Parallel RapidIO Intconnect Specification*, I/O read home and flush with data transactions require special handling. Outbound I/O 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 I/O read home transactions. Therefore, regardless of the size of the original request, the address for outbound I/O 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 controller. These transactions must map to an ATMU window in which the ROWAR*n* 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.

To comply with the *Parallel RapidIO Intconnect 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 be at least a double word, and atomic transactions may only request 1, 2, or 4 bytes. Furthermore, atomic transactions must meed the RapidIO alignment requirements as specified in the *Parallel RapidIO Intconnect Specification*.

The *Parallel RapidIO Intconnect 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.

The LCSBA1CSR window does not have specific RIWBAR, RIWTAR or RIWAR registers. However, for the purposes of translation, bits 1–14 of the LCSBA1CSR correspond to bits 10–23 of RIWBAR*n*, and bits 12–23 of the CCSRBAR correspond to bits 12–23 of RIWTAR*n*. The hard-coded attributes for this window as they correspond to the RIWAR register fields are: EN = 0b1, TGINT = 0b1111, RDTYP and WRTYP = 0b0100 and SIZE = 0b010011.

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 cleared. Similarly, the no_snoop bit is set according to the corresponding programming model of the specific transaction type regardless of the value specified in the RIWAR of the selected ATMU window. For more information about these transactions, see 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 to comply with the *Parallel RapidIO Intconnect Specification*. For example, inbound I/O 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, I/O read home transactions can request less data than the maximum of 32 bytes.

Inbound reads that target the PCI controller should have a priority of zero as a special requirement. 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.

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 *Parallel RapidIO Intconnect Specification* and must target the DDR memory controller. Inbound atomic transactions targeting interfaces other than the DDR memory controller result in boundedly undefined behavior. Also, all inbound maintenance transactions must have a size of four bytes.

The *Parallel RapidIO Intconnect Specification* should be referenced for any additional requirements that are not listed in this section.

16.7.2.4 ATMU Boundary Crossing Errors

During address translation and mapping, some transactions may generate an error condition and signal an interrupt. To prevent inbound and outbound transactions from corrupting local memory space, ATMU boundary crossing errors are detected. An error is detected when an inbound or outbound transaction has a data payload that crosses its selected ATMU window boundary. An error is also detected when the given transaction’s data payload crosses into the space of a higher-priority window. The settings of the port notification/fatal error register set described in Section 16.3.2.3, “Error Management Registers,” determine if the error condition is detected and if PNFEDR[AXE] is set. The settings in this register set also determine if an interrupt is signalled upon detection of the error condition.

16.8 RapidIO Message Unit

This description is an extension of Section 16.7.1, “Outbound ATMU Translation,” and Section 16.7.2, “Inbound ATMU Translation.” 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 in the *Parallel RapidIO Intconnect Specification*.

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 controller places the message in a queue located in main (DDR) 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 message 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 processor core. When a doorbell is received, an interrupt (if enabled) is communicated to the processor.

The most common use of the message passing model is in systems where a processing element can only access memory that is local to itself, and communication between processing elements is achieved through message passing and communication is address independent.

There are two types of RapidIO 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.

Message and doorbell controllers are controlled through run-time registers described in Section 16.3.3.1, "RapidIO Outbound Message Registers," Section 16.3.3.2, "RapidIO Inbound Message Registers," and Section 16.3.3.3, "RapidIO Doorbell Registers."

16.8.2 Message Unit Features

The message unit contains the following features:

- One inbound data message structure (inbox)
- One outbound data message structure (outbox)
- Support for chaining and direct modes in the outbox
- Support for up to 16 segments per message
- Support for up to 256 bytes per packet, and up to 4 Kbytes of data per message
- Support for one inbound doorbell message structure
- Support for receiving messages into any mailbox, any letter
- Support for transmitting messages from any mailbox

16.8.3 Message Unit Modes of Operation

The message unit operates in the following modes:

- 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 describes controller operations, formats, interrupts, and response conditions.

16.8.4.1 Data Message Controller

The RapidIO controller 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 controller 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 processor core 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 RapidIO data message controller 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 one letter (0) in the 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 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 message unit 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 that 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.
- An end of message interrupt is 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, and 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. 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 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 ODQDPAR to point to the next descriptor in the queue and repeats this process until the enqueue and dequeue pointers are equal.

Figure 16-82 shows 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 processor core adds descriptors to the head of the queue and the outbox message controller takes them off from the tail.

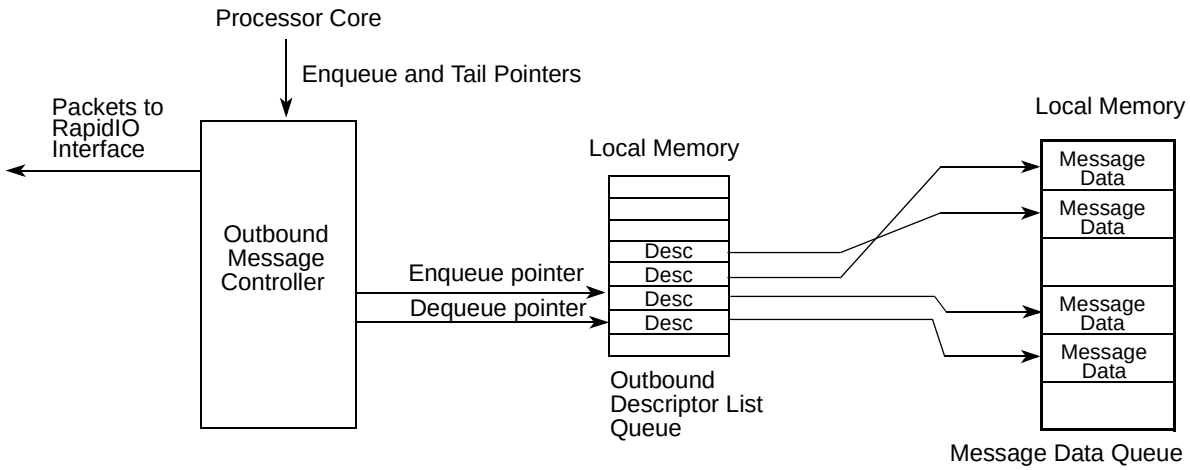


Figure 16-82. Outbound Message Queue Structure

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:

- Initialize ODQDPAR and ODQEPAR 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 ODQDPAR to be saved 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 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 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 while the message unit is busy.

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 ODQDPAR to point to the first descriptor in memory and ODQEPAR and 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 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 because software directly controls the enqueue pointer; therefore, 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 ODQEPAR 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, hardware tracks queue fullness in normal chaining mode and reports an interrupt if the queue becomes full. After the interrupt bit is set, it is the responsibility of the software to clear the interrupt condition by setting OSR[QFI]. QFI is not cleared with the write if the queue is still full. When 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 following process adds a descriptor to the circular memory queue in normal mode:

- Ensure that the circular queue is not full by 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 after 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 then set, the message unit is re-initialized to chaining mode and the outbound descriptor queue dequeue pointer address is saved 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 begins 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] must be cleared and then set. This has 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

Message descriptors contain 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 16-103 describes the outbound message unit descriptor.

Table 16-103. Outbound Message Unit Descriptor Summary

Offset	Descriptor Field	Description
0x00	Reserved	—
0x04	Source address	Source address of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the source address register.
0x08	Destination port	Destination port of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the destination port register.
0x0C	Destination attributes	Transaction attributes of the message operation. After the message controller reads the descriptor from memory, this field is loaded into the destination attributes register.
0x10	Reserved	—
0x14	Reserved	—
0x18	Double-word count	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.
0x1C	Reserved	—

Figure 16-83 shows the queue dequeue pointer and an associated descriptor. The descriptor is only valid if the enqueue and dequeue pointers are not equal.

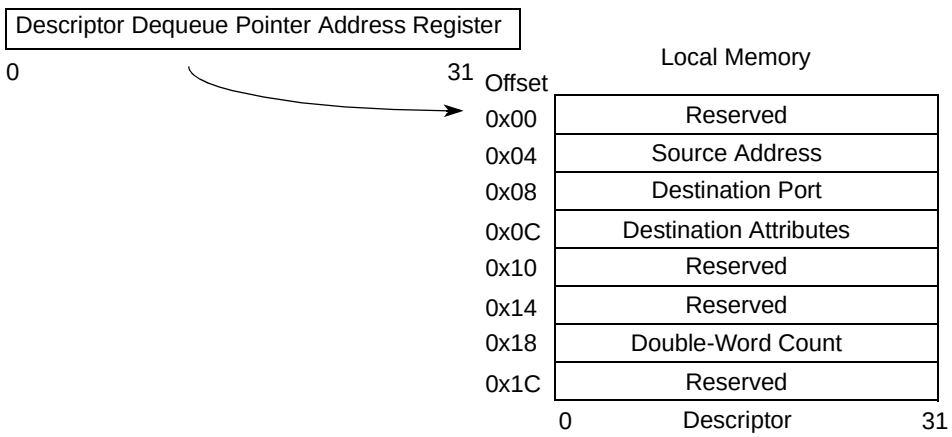


Figure 16-83. Descriptor Dequeue Pointer and Descriptor

16.8.4.2.5 Outbox Controller Interrupts

The outbox generates the following four interrupts, which can be individually enabled.

- Queue overflow interrupt. 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. Generated if the enqueue pointer catches up to the dequeue pointer, the queue is not empty, and the interrupt is enabled. OSR[QFI] is set if OMR[QFIE] is set.
- Queue empty interrupt. 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. 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 prevents the recipient from using data in the message. For example, if the outbox controller reads message data from its DRAM memory and 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, the inbox controller 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 software controls the dequeue pointer.

Figure 16-84 shows 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 processor core takes them off from the tail. After processing a message, the processor core 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.

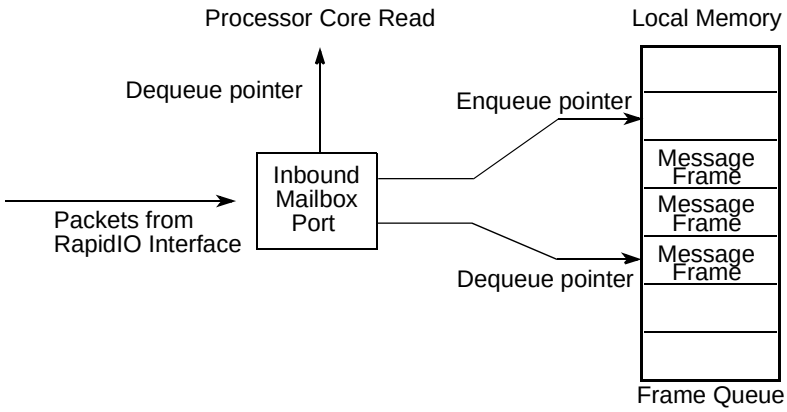


Figure 16-84. Inbound Mailbox Structure

The sequence of events that takes 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.
- After 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

The following two conditions generate a logical layer retry (response retry):

- Inbox receives a message but 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) as follows:

- Inbox receives a message and it is not enabled
- Inbox is in an error state caused by a previous message
- Inbox receives a segment that exceeds the message length (msgseg > msglen)

- Inbox receives a message segment 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 the following 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 describes 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

The RapidIO protocol supports a doorbell message type that contains no data payload. Even though the RapidIO architecture references outbound and inbound doorbell message controllers, the RapidIO controller supports only an inbound doorbell message queue. Outbound doorbells are generated outside RapidIO within the outbound ATMU. 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 16-85 shows an example of the structure of the inbound doorbell queue and its pointers. The doorbell queue of the RapidIO controller has eight entries, three of which are currently valid. The doorbell controller adds doorbell information to the head of the queue and the processor core removes it 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.

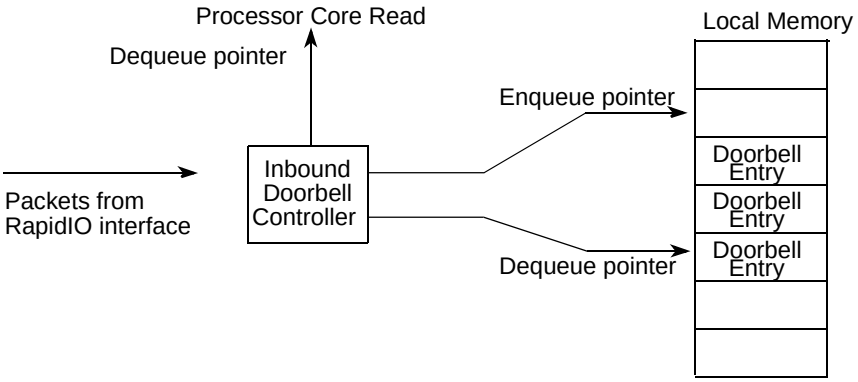


Figure 16-85. Inbound Doorbell Queue and Pointer Structure

16.8.4.4.1 Inbound Doorbell Reception

The following sequence of events occurs during reception of a doorbell message:

- The inbox receives a doorbell 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 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 32-bit offsets, one for target information and one for source information. Table 16-104 shows the target information.

Table 16-104. Target Information Definition

Bits	Name	Description
0–23	—	Reserved
24–31	TID	Target ID field from the received doorbell packet

Table 16-105 shows the source information.

Table 16-105. Source Information Definition

Bits	Name	Description
0–7	—	Reserved
8–15	SID	Source ID field from the received doorbell packet
16–31	INFO	Information field from the received doorbell packet

Figure 16-86 shows the doorbell queue entry fields and their related offsets.

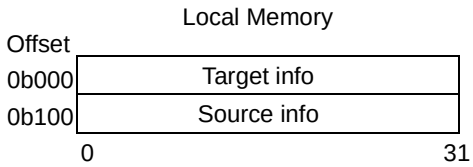


Figure 16-86. Doorbell Entry Format

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 the following two different interrupts that can be individually enabled:

- Doorbell-in-queue interrupt. 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. 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-free device to a control processor or other system host.

Figure 16-87 shows 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 processor core must explicitly 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.

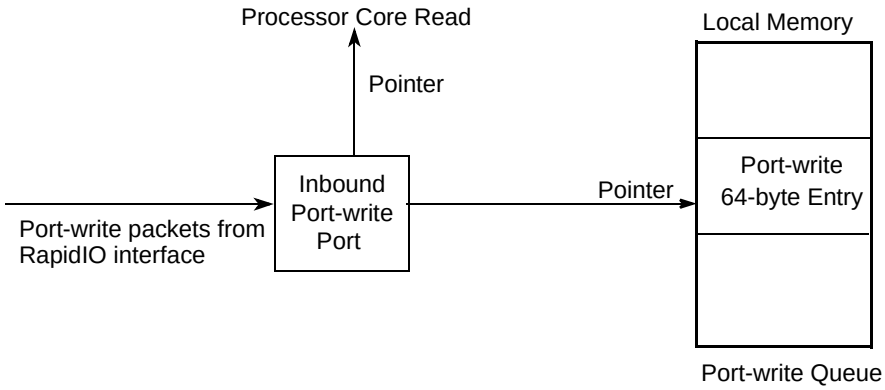


Figure 16-87. Inbound Port-Write Structure

Upon successfully receiving a port-write transaction, if PWMR[QFIE] is set, PWSR[QFI] is set and an interrupt is generated to the processor core.

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-on reset.
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 inter-operability portion of the *Parallel RapidIO Intconnect Specification* for configuring the RapidIO system.

Part IV
Global Functions and Debug

Part IV defines other global blocks of the MPC8560. The following chapters are included:

- 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

This chapter describes the global utilities of the MPC8560. It provides signal descriptions, register descriptions, and a functional description of these utilities.

17.1 Overview

The global utilities block controls power management, I/O device enabling, power-on reset (POR) configuration monitoring, general-purpose I/O signal configuration, alternate function selection for multiplexed signals, and clock control.

17.2 Global Utilities Features

This section provides an overview of global utilities features.

17.2.1 Power Management and Block Disables

The following features 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 block disables)

17.2.2 Accessing Current POR Configuration Settings

The POR configuration values of all device parameters sampled from pins at reset are available through memory-mapped registers in the global utilities block.

17.2.3 General-Purpose I/O

The PCI and TSEC2 data bus signals can be used as general-purpose I/O signals when not used for their primary function. Memory-mapped registers in the global utilities block provide control and status for the use of these signals. A general purpose input register is loaded with the values of the local bus address/data pins at the negation of $\overline{\text{HRESET}}$.

17.2.4 Interrupt and Local Bus Signal Multiplexing

IRQ[9:11] and $\overline{\text{LCS}}$[5:7] serve multiple functions that can be selected by configuration registers in the global utilities block.

The multiplexing of the CPM signals occurs through the CPM programming model. See Chapter 45, “Parallel I/O Ports,” for details on CPM signal multiplexing.

17.2.5 Clock Control

The global utilities block also selects the internal clock signal driven on CLK_OUT.

17.3 External Signal Descriptions

The following subsections provide information about signals that serve as global utilities.

17.3.1 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	Description	Reference (Section/Page)
ASLEEP	O	Signals that the device has reached a sleep state	17.5.1.5.3/17-24
$\overline{\text{CKSTP\_IN}}$	I	Checkstop input	—
$\overline{\text{CKSTP\_OUT}}$	O	Checkstop output	—
CLK_OUT	O	Clock out. Selected by CLKOCR values	17.4.1.16/17-18

17.3.2 Detailed Signal Descriptions

Table 17-2 describes signals in the global utilities block in detail.

Table 17-2. Detailed Signal Descriptions

Signal	I/O	Description	
ASLEEP	O	Asleep. See Section 17.5.1.5.3, “Sleep Mode.” After negation of $\overline{\text{HRESET}}$, ASLEEP is asserted until the device completes its power-on reset sequence and reaches its ready state.	
		State Meaning	Asserted—Indicates that the device is either still in its power-on reset sequence or has reached a sleep state after a power-down command is issued by software Negated—The device is not in sleep mode. (It has either awakened from a power-down state or completed the POR sequence.)
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks Negation—Negates synchronously with SYSCLK when leaving power-on sequence; otherwise negation is asynchronous

Table 17-2. Detailed Signal Descriptions (continued)

Signal	I/O	Description	
CKSTP_IN	I	Checkstop in	
		State Meaning	Asserted—Indicates that the e500 core must enter a hard stop condition. All e500 clocks are turned off. CKSTP_OUT is asserted. The rest of MPC8560 device logic, including memory controllers, internal memories and registers, and I/O interfaces, remains functional. Negated—Indicates that normal operation should proceed
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks Negation—Must remain asserted until the MPC8560 is reset with assertion of HRESET
CKSTP_OUT	O	Checkstop out	
		State Meaning	Asserted—Indicates that the e500 core of the MPC8560 is in a checkstop state. The rest of the MPC8560 logic remains functional. Negated—Indicates normal operation. After CKSTP_OUT has been asserted, it is negated after the next negation (low-to-high transition) of HRESET.
		Timing	Assertion—May occur at any time; may be asserted asynchronously to the input clocks Negation—Must remain asserted until the device has been reset with a hard reset
CLK_OUT	O	Clock out. Reflects clock signal selected by CLKOCR (see Section 17.4.1.16, “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]

17.4 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	PORPLLSR—POR PLL ratio status register	R	0x00*nn*_00*nn*	17.4.1.1/17-4
0xE_0004	PORBMSR—POR boot mode status register	R	0x*nnnn*_0000	17.4.1.2/17-5
0xE_0008	PORIMPSCR—POR I/O impedance status and control register	R/W	0x000*n*_007F	17.4.1.3/17-6
0xE_000C	PORDEVSR—POR I/O device status register	R	See ref.	17.4.1.4/17-7
0xE_0010	PORDBGMSR—POR debug mode status register	R	See ref.	17.4.1.5/17-9
0xE_0020	GPPORCR—General-purpose POR configuration register	R	See ref.	17.4.1.6/17-9
Signal Multiplexing and GPIO Controls				
0xE_0030	GPIOCR—GPIO control register	R/W	0x0000_0000	17.4.1.7/17-10
0xE_0040	GPOUTDR—General-purpose output data register	R/W	0x0000_0000	17.4.1.8/17-11

Table 17-3. Global Utilities Block Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0xE_0050	GPINDR—General-purpose input data register	R	0x*nnnn*_0000	17.4.1.9/17-12
0xE_0060	PMUXCR—Alternate function signal multiplex control	R/W	0x0000_0000	17.4.1.9/17-12
Device Disables				
0xE_0070	DEVDISR—Device disable control	R/W	0x0000_0000	17.4.1.11/17-13
Power Management Registers				
0xE_0080	POWMGTCSR—Power management status and control register	R/W	0x0000_0000	17.4.1.12/17-15
Interrupt Reporting				
0xE_0090	MCPSUMR—Machine check summary register	Read/Clear	0x0000_0000	17.4.1.13/17-16
Version Registers				
0xE_00A0	PVR—Processor version register	R	e500 processor version	17.4.1.14/17-17
0xE_00A4	SVR—System version register	R	MPC8560 system version	17.4.1.15/17-18
Debug Control				
0xE_0E00	CLKOCR—Clock out select register	R/W	0x0000_0000	17.4.1.16/17-18
0xE_0E10	DDRDLLCR—DDR DLL control register	R/W	0x0000_0000	17.4.1.17/17-19
0xE_0E20	LBDLLCR—LBC DLL control register	R/W	0x0000_0000	17.4.1.18/17-20

17.4.1 Register Descriptions

This section describes the global utilities registers in detail.

17.4.1.1 POR PLL Status Register (PORPLLSR)

PORPLLSR, shown in Figure 17-1, contains the settings for the PLL ratios as set by the cfg_sys_pll[0:3] and cfg_core_pll[0:1] POR configuration pins. See Section 4.4.3.1, “System PLL Ratio,” and Section 4.4.3.2, “e500 Core PLL Ratio,” for more information.

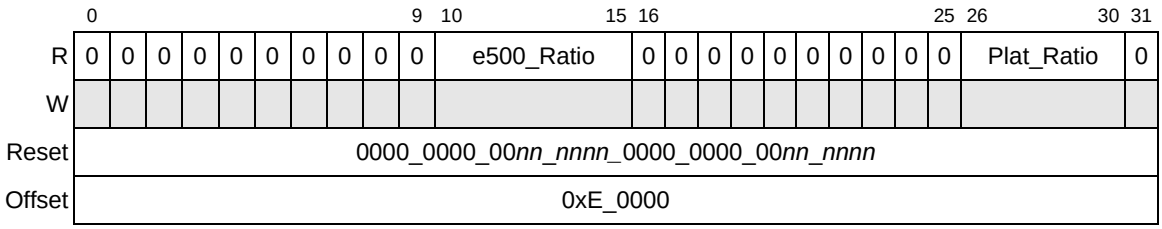


Figure 17-1. POR PLL Status Register (PORPLLSR)

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. The 2 lsbs of this field correspond to the values on cfg_core_pll[0:1] at the negation of HRESET. Patterns not shown are reserved. 000100 2:1 000101 5:2 000110 3:1 000111 7:2
16–25	—	Reserved
26–30	Plat_Ratio	Clock ratio between the CCB (platform) clock and SYSCLK. The 4 lsbs correspond to the values on cfg_sys_pll[0:3] at the negation of HRESET. Patterns not shown are reserved. 00010 2:1 — 01000 8:1 00011 3:1 — 01001 9:1 00100 4:1 — 01010 10:1 00101 5:1 — 01100 12:1 00110 6:1 — 10000 16:1
31	—	Reserved

17.4.1.2 POR Boot Mode Status Register (PORBMSR)

The PORBMSR, shown in Figure 17-2, reports setting of the POR configuration pins that control the boot mode settings (described in Section 4.4.3.3, “Boot ROM Location,” Section 4.4.3.5, “CPU Boot Configuration,” and Section 4.4.3.6, “Boot Sequencer Configuration,”) and the default settings of RapidIO and PCI host/agent modes (described in Section 4.4.3.4, “Host/Agent Configuration”).

	0	1			4	5		7	8	9	10		11	12	13	14		15	16													31
R	BCFG	0	0	0	0	ROM_LOC			0	0	BSCFG			0	0	HA			0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	*n*000_0*nn*n_00*nn*_00*nn*_0000_0000_0000_0000																															
Offset	0xE_0004																															

Figure 17-2. POR Boot Mode Status Register (PORBMSR)

For information about how the RapidIO interface is configured, see Section 16.3.1.19, “Port General Control Command and Status Register (PGCCSR).” For more information about the PCI configurations, see Section 15.3.2.19, “PCI Bus Function Register (PBFR).” 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

17.4.1.3 POR I/O Impedance Status and Control Register (PORIMPSCR)

PORIMPSCR, shown in Figure 17-3,contains the current I/O driver impedances for local bus and PCI interfaces.

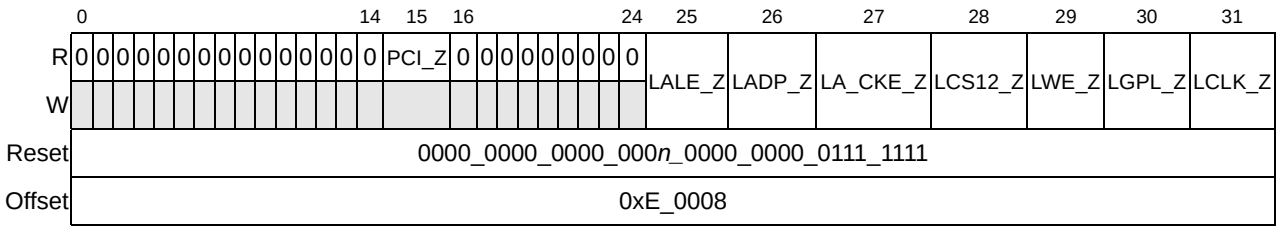


Figure 17-3. POR I/O Impedance Status and Control Register (PORIMPSCR)

The I/O impedance of local bus signals (including the local bus clock) is controlled through this register. The I/O impedance of PCI signals is controlled by POR configuration pins (described in

Section 4.4.3.13, “PCI I/O Impedance.”) The *MPC8560 Integrated Processor Hardware Specifications* provides exact I/O impedances.

Table 17-6 describes PORIMPSCR fields.

Table 17-6. PORIMPSCR Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	PCI_Z	PCI/PCI-X I/O impedance 0 Low impedance 1 High impedance
16–24	—	Reserved
25	LALE_Z	I/O impedance for local bus address latch enable 0 Low impedance 1 High impedance
26	LADP_Z	I/O impedance for local bus address/data and data parity (LAD[0:31] and LDP[0:7]) 0 Low impedance 1 High impedance
27	LA_CKE_Z	I/O impedance for local bus address and clock enable (LA[27:31] and LCKE) 0 Low impedance 1 High impedance
28	LCS12_Z	I/O impedance for two local bus chip selects (LCS[1] and LCS[2] only). NOTE: Other chip selects use a fixed high I/O impedance 0 Low impedance 1 High impedance
29	LWE_Z	I/O impedance for local bus write enables (LWE[0:3]) 0 Low impedance 1 High impedance
30	LGPL_Z	I/O impedance for local bus general-purpose lines (LGPL[0:5]) 0 Low impedance 1 High impedance
31	LCLK_Z	I/O impedance for local bus clocks (LCLK[0:2]) 0 Low impedance 1 High impedance

17.4.1.4 POR Device Status Register (PORDEVSR)

Shown in Figure 17-4, PORDEVSR reports other POR settings for I/O devices as described in Section 4.4.3.7, “TSEC Width,” Section 4.4.3.8, “TSEC1 Protocol,” Section 4.4.3.9, “TSEC2 Protocol,” Section 4.4.3.16, “PCI-X Configuration,” Section 4.4.3.14, “PCI Arbiter Configuration,” Section 4.4.3.12, “PCI Width Configuration,” Section 4.4.3.10, “RapidIO Transmit Clock Source,” and Section 4.4.3.11, “RapidIO Device ID.”

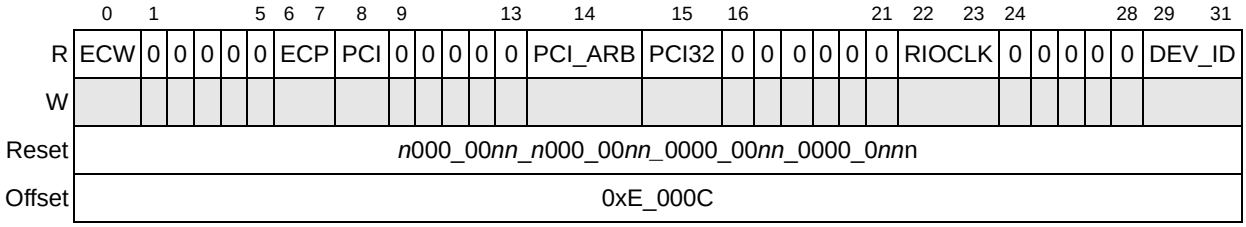


Figure 17-4. POR Device Status Register (PORDEVSR)

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
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)
24–28	—	Reserved
29–31	DEV_ID	RapidIO device ID [5:7]

17.4.1.5 POR Debug Mode Status Register (PORDBGMSR)

PORDBGMSR, shown in Figure 17-5, holds debug mode settings from the POR configuration pins as described in Section 4.4.3.17, "Memory Debug Configuration," Section 4.4.3.18, "DDR Debug Configuration," and Section 4.4.3.15, "PCI Debug Configuration."

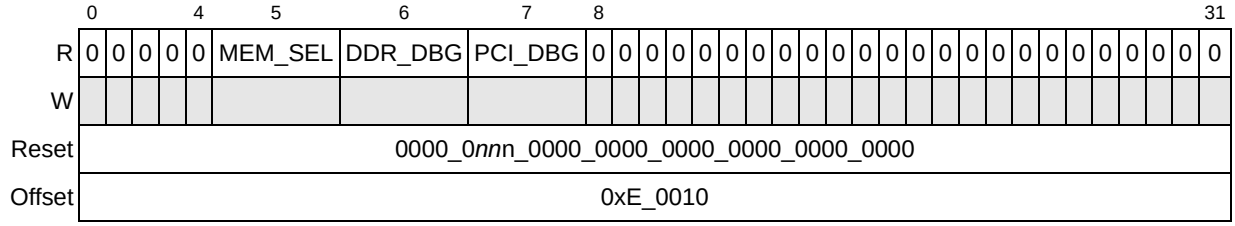


Figure 17-5. POR Debug Mode Status Register (PORDBGMSR)

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 controller is driving MSRCID[0:4] and MDVAL 0 Local bus controller is driving debug information 1 DDR SDRAM controller is driving debug information
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 PCI_AD[62:58]. 1 PCI/PCI-X drives address/data 62:58 onto PCI_AD[62:58].
8–31	—	Reserved

17.4.1.6 General-Purpose POR Configuration Register (GPPORCR)

Shown in Figure 17-6, GPPORCR stores the value sampled from the local bus address/data signals, LAD[0:31], during POR, as described in Section 4.4.3.21, "General-Purpose POR Configuration." Software can use this value to inform the operating system about initial system configuration. Typical interpretations include circuit board type, board ID number, or a list of available peripherals.

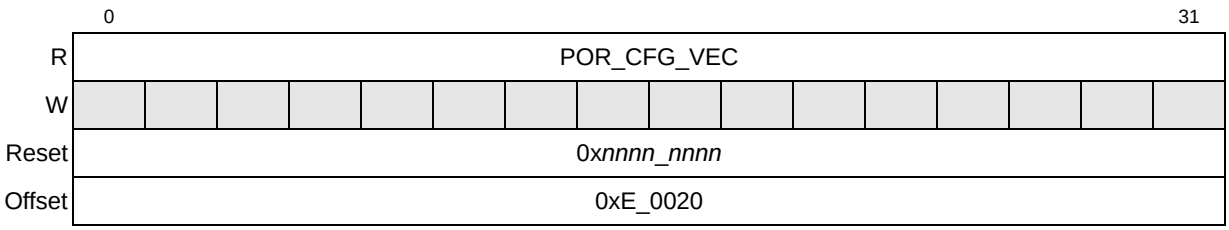


Figure 17-6. POR Configuration Register (GPPORCR)

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 HRESET. Note that if nothing is driven on these signals during reset, the value of this register is indeterminate.

17.4.1.7 General-Purpose I/O Control Register (GPIOCR)

Shown in Figure 17-7, GPIOCR contains the enable bits for each group of pins that may be used for general-purpose I/O. These bits have meaning only if the pins are not being used for their primary function. Note that when these signals are enabled as general-purpose I/O signals, they are read and written through GPINDR and GPOUTDR described in Section 17.4.1.9, "General-Purpose Input Data Register (GPINDR)," and Section 17.4.1.8, "General-Purpose Output Data Register (GPOUTDR)." Section 17.5.2, "General-Purpose I/O Signals," describes the use of general-purpose I/O signals.

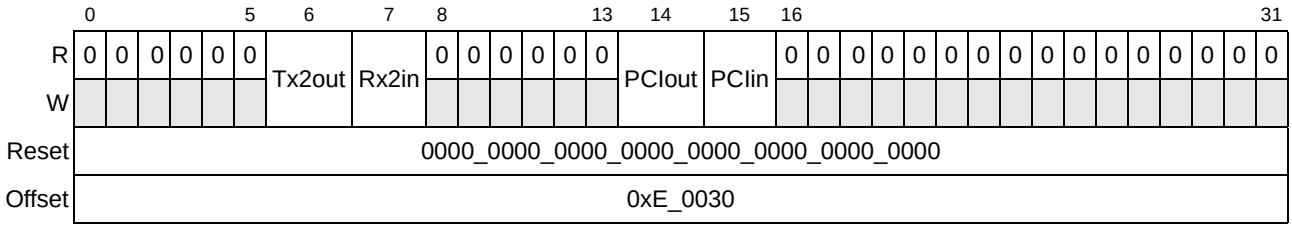


Figure 17-7. General-Purpose I/O Control Register (GPIOCR)

Table 17-10 describes the bit settings of GPIOCR.

Table 17-10. GPIOCR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	Tx2out	TSEC2_Tx[0:7] output enable. Used to enable TSEC2_Tx[0:7] for use as general-purpose output if the TSEC2 interface is disabled 0 TSEC2_Tx[0:7] is not used as general-purpose output. 1 TSEC2_Tx[0:7] is used as general-purpose output.
7	Rx2in	TSEC2_Rx[0:7] input enable. Used to enable TSEC2_Rx[0:7] for use as general-purpose input if the TSEC2 interface is disabled 0 TSEC2_Rx[0:7] is not used as general-purpose input. 1 TSEC2_Rx[0:7] is used as general-purpose input.
8–13	—	Reserved
14	PCIout	PCI_AD[47:40] output enable. Used to enable PCI_AD[47:40] for use as general-purpose output if PCI interface is in 32-bit mode or is disabled 0 PCI_AD[47:40] is not used as general-purpose output. 1 PCI_AD[47:40] is used as general-purpose output.

Table 17-10. GPIOCR Field Descriptions (continued)

Bits	Name	Description
15	PCIin	PCI_AD[39:32] input enable. Used to enable PCI_AD[39:32] for use as general-purpose input if the PCI interface is in 32-bit mode or is disabled 0 PCI_AD[39:32] is not used as general-purpose input. 1 PCI_AD[39:32] is used as general-purpose input.
16–31	—	Reserved

17.4.1.8 General-Purpose Output Data Register (GPOUTDR)

GPOUTDR, shown in Figure 17-8, contains the data driven as general-purpose output on TSEC2_TxD[0:7] and/or PCI_AD[47:40] when either of these buses is configured as general-purpose I/O buses, as described in Section 17.4.1.7, “General-Purpose I/O Control Register (GPIOCR).” Writes to GPOUTDR affect only pins enabled as general-purpose outputs. Reads return valid data only for bits corresponding to pins enabled as general-purpose outputs. GPOUTDR may be accessed using single byte writes (using big-endian addressing) so that writes to one byte do not affect outputs controlled by others.

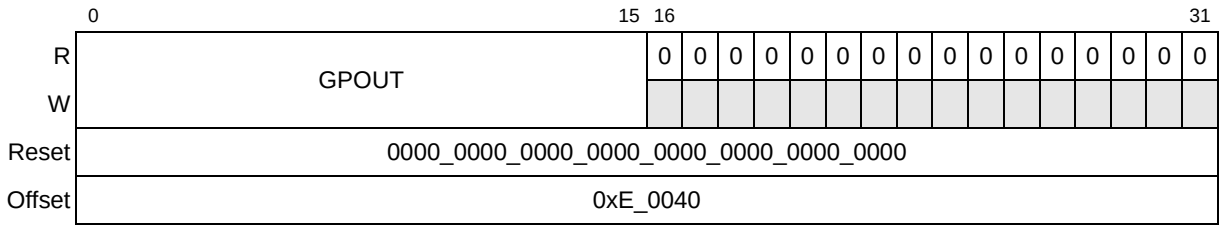


Figure 17-8. General-Purpose Output Data Register (GPOUTDR)

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

17.4.1.9 General-Purpose Input Data Register (GPINDR)

GPINDR, shown in Figure 17-9, contains the data currently sampled as general-purpose input on TSEC2_RxD[0:7] and/or PCI_AD[39:32] when either of these buses is configured as general-purpose I/O buses in GPIOCR (see Section 17.4.1.7, "General-Purpose I/O Control Register (GPIOCR)"). GPINDR bits are updated only if the associated bits are configured as general-purpose input pins rather than their primary functions.

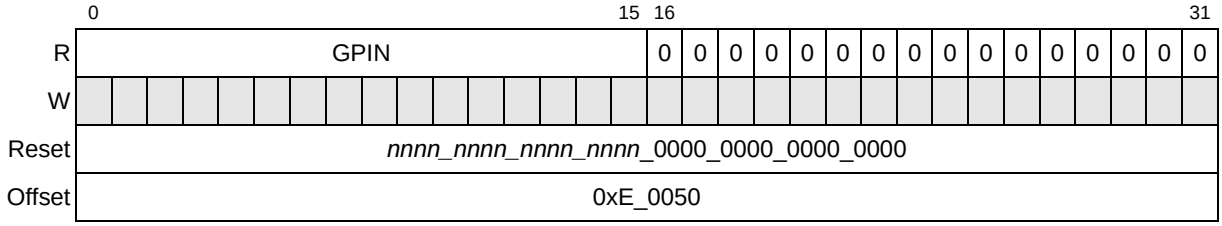


Figure 17-9. General-Purpose Output Data Register (GPINDR)

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

17.4.1.10 Alternate Function Signal Multiplex Control Register (PMUXCR)

Shown in Figure 17-10, PMUXCR contains bits that enable DMA channels 2 and 3 which exist as alternate functions on local bus chip select pins $\overline{\text{LCS}}$[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 45, "Parallel I/O Ports."

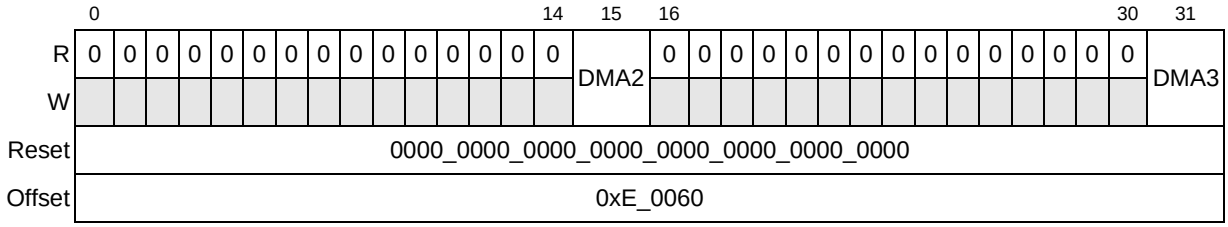


Figure 17-10. Alternate Function Pin Multiplex Control Register (PMUXCR)

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 functions as DMA_DREQ2 LCS6 functions as DMA_DACK2 LCS7 functions as 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 functions as DMA_DREQ3 IRQ10 functions as DMA_DACK3 IRQ11 functions as DMA_DDONE3

17.4.1.11 Device Disable Register (DEVDISR)

DEVDISR, shown in Figure 17-11, contains disable bits for various MPC8560 functional blocks.

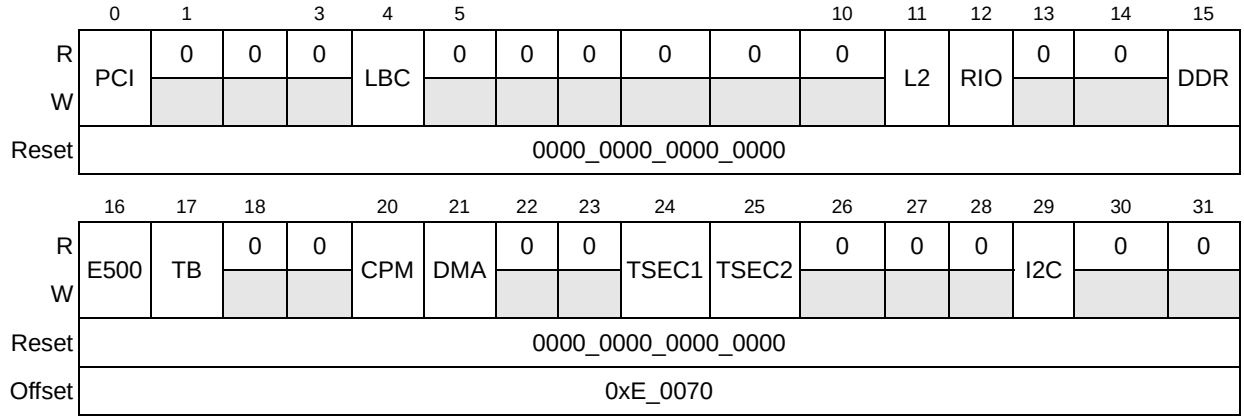


Figure 17-11. Device Disable Register (DEVDISR)

All functional blocks are enabled after reset; unneeded blocks can be disabled to reduce power consumption or allow their signals to be used as general-purpose I/O signals. See Section 17.4.1.7, "General-Purpose I/O Control Register (GPIOCR)." Blocks disabled by DEVDISR must not be re-enabled without a hard reset. Section 17.5.1.4, "Shutting Down Unused Blocks," has more information on the use of DEVDISR. Table 17-14 describes DEVDISR fields.

Table 17-14. DEVDISR Field Descriptions

Bits	Name	Description
0	PCI	PCI/PCI-X controller disable 0 PCI/PCI-X controller enable 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 enable 1 Local bus controller disable
5–10	—	Reserved
11	L2	Level-2 cache disable 0 L2 cache enable 1 L2 cache disable
12	RIO	RapidIO controller disable 0 RapidIO controller enable 1 RapidIO controller disable, including RapidIO messaging, doorbell, and port write controllers.
13–14	—	Reserved
15	DDR	DDR SDRAM controller disable 0 DDR SDRAM controller enable 1 DDR SDRAM controller disable
16	E500	e500 core disable 0 e500 core enable 1 e500 core disable. Places the core in the core-stopped state in which it does not respond to interrupts. Equivalent to nap mode. Instruction fetching is stopped, snooping is disabled, and clocks are shut down to all functional units of the core except for the timer facilities. For more information, see Section 17.5.1.4, "Shutting Down Unused Blocks."
17	TB	Time base (timer facilities) of the e500 core disable 0 Timer facilities enabled 1 Timer facilities disabled
18–19	—	Reserved
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

Table 17-14. DEVDISR Field Descriptions (continued)

Bits	Name	Description
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

17.4.1.12 Power Management Control and Status Register (POWMGTCSR)

Shown in Figure 17-12, POWMGTCSR contains bits for placing the MPC8560 into low power states and for controlling when it wakes up. It also contains power management status bits. See Section 17.5.1.8.2, "Interrupts and Power Management Controlled by POWMGTCSR," for more information.

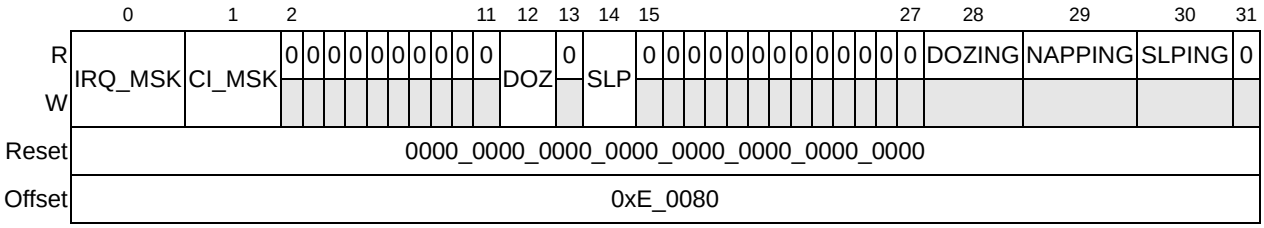


Figure 17-12. Power Management Control and Status Register (POWMGTCSR)

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.
2–11	—	Reserved

Table 17-15. POWMGTCSR Field Descriptions (continued)

Bits	Name	Description
12	DOZ	Doze mode 0 No request to put device in doze mode. Note that this bit is automatically cleared on MCP, UDE, SRESET, *core_tbint* (from the core) and also *int* and *cint* if not masked. 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 HID0[DOZE].
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, snooping of L1 caches is disabled, and most functional blocks are shut down in both the e500 core and the system logic.
15–27	—	Reserved
28	DOZING	Doze status 0 Device is not in doze mode 1 The MPC8560 is in doze mode because POWMGTCSR[DOZ] is set or because HID0[DOZE] and MSR[WE] (in the e500 core) are set. The core has halted instruction fetching, but all other functional blocks in the core and device are running.
29	NAPPING	Nap status 0 Device is not in nap mode. 1 The MPC8560 is in nap mode because HID0[NAP] and MSR[WE] are set. The core has halted instruction fetching, snooping of the L1 caches is disabled, and all of the core's functional units except the timer facilities are shut down. All functional blocks in the device are running.
30	SLPING	Sleep status 0 Device is not attempting to reach sleep mode. 1 The device is attempting to SLEEP because POWMGTCSR[SLP] is set or because HID0[SLEEP] and MSR[WE] (in the e500 core) are set. Most functional blocks in the core and device are shut down or are attempting to shut down.
31	—	Reserved. Should be cleared.

17.4.1.13 Machine Check Summary Register (MCPSUMR)

Shown in Figure 17-13, MCPSUMR contains bits summarizing some of the sources of a pending machine check interrupt. All MCPSUMR bits function as write-one-to-clear.

NOTE

Register fields designated as write-one-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

Note that other conditions can cause a machine check condition not summarized in MCPSUMR. For example, uncorrectable read errors cause the assertion of *core_fault_in*, which may directly cause a machine check (if HID1[RFXE] = 1). If RFXE = 0, the assertion of *core_fault_in* does not directly cause a machine check interrupt, but must be handled by the block that generated the error. For more information about RFXE, see Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)."

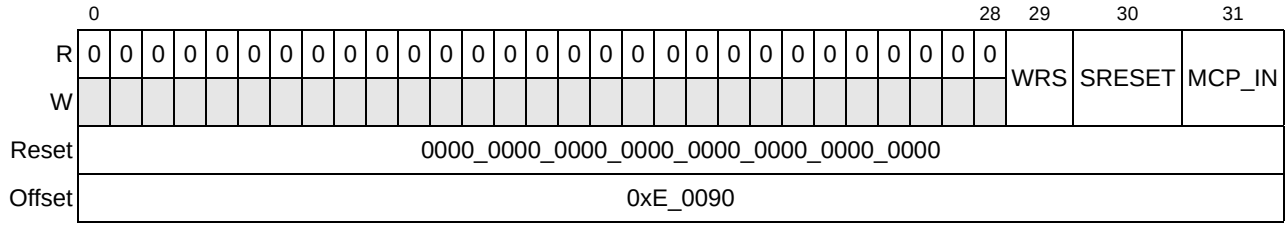


Figure 17-13. Machine Check Summary Register (MCPSUMR)

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 machine check 0 Machine check exception was not caused by watchdog timer 1 Machine check was caused by a soft reset condition from the e500 watchdog timer as configured in the core's TSR. Specifically, TSR[WRS] = 01 and a watchdog reset condition occurred.
30	SRESET	Soft reset machine check 0 Machine check exception was not caused by $\overline{\text{SRESET}}$ assertion 1 Machine check exception was caused by the assertion of the $\overline{\text{SRESET}}$ input signal
31	MCP_IN	$\overline{\text{MCP}}$ signal asserted 0 Machine check exception was not caused by $\overline{\text{MCP}}$ assertion 1 Machine check exception was caused by the assertion of the $\overline{\text{MCP}}$ input signal

17.4.1.14 Processor Version Register (PVR)

Shown in Figure 17-14, the PVR contains the e500 processor version number. It is a memory-mapped copy of the PVR in the e500 core (and is therefore accessible to external devices). See Section 6.5.3, "Processor Version Register (PVR)." Section 5.2, "e500 Processor and System Version Numbers," lists the complete values for the MPC8560.

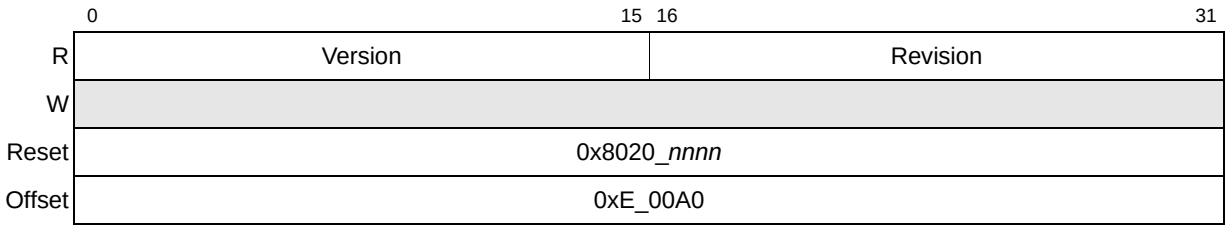


Figure 17-14. Processor Version Register (PVR)

Table 17-17 describes the fields of PVR.

Table 17-17. PVR Field Descriptions

Bits	Name	Description
0–15	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.
16–31	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.

17.4.1.15 System Version Register (SVR)

Shown in Figure 17-15, the SVR contains the system version number for the MPC8560 implementation. This value can also be read though the SVR SPR of the e500 core. See Section 6.5.4, "System Version Register (SVR)." Section 5.2, "e500 Processor and System Version Numbers," lists the complete values for the MPC8560.

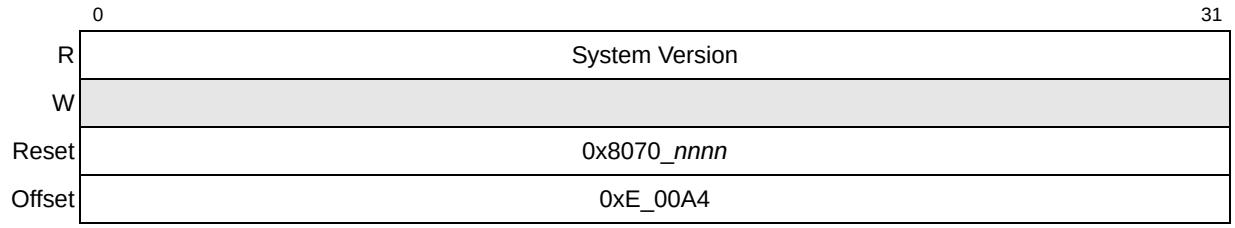


Figure 17-15. System Version Register (SVR)

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

17.4.1.16 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.

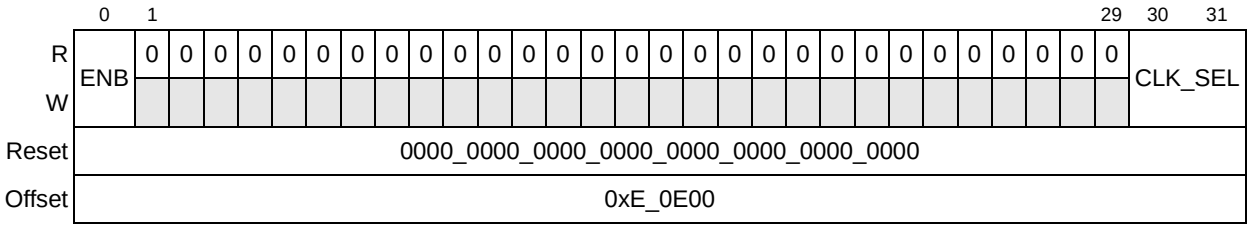


Figure 17-16. Clock Out Control Register (CLKOCR)

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 (echoes SYSCLK input) 11 SYSCLK divided by 2 (demonstrates platform PLL lock)

17.4.1.17 DDR DLL Control Register (DDRDLLCR)

The DDRDLLCR, shown in Figure 17-17, contains bits that allow control and debug of the DDR SDRAM controller's DLL. The DLL delay chain consists of 128 tap points.

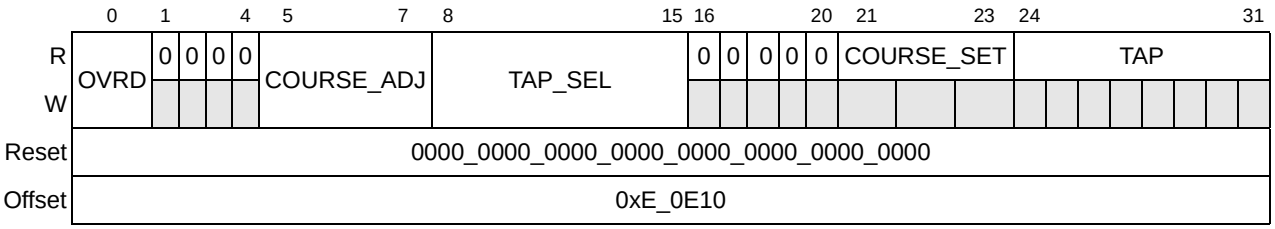


Figure 17-17. DDR DLL Control Register (DDRDLLCR)

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

Table 17-20. DDRDLLCR Field Descriptions (continued)

Bits	Name	Description
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

17.4.1.18 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.

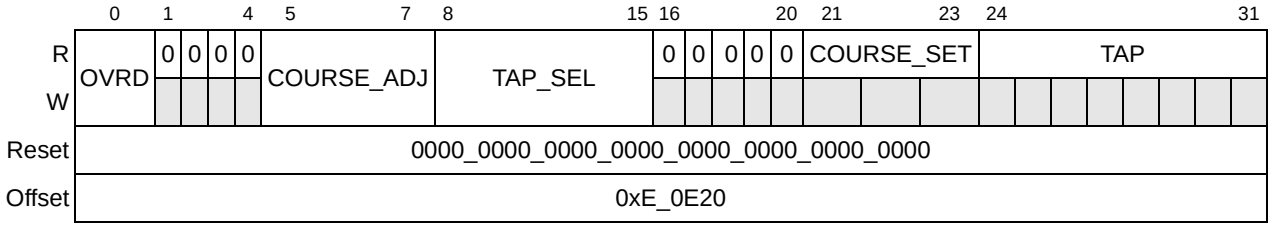


Figure 17-18. Local Bus DLL Control Register (LBDLLCR)

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

17.5 Functional Description

This section describes the global utilities from a functional perspective.

17.5.1 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 through a memory-mapped register (DEVDISR). Additionally, software running on the e500 core can access the core's SPRs to put the device into doze, nap, or sleep power down state. Finally, software can access a memory-mapped register (POWMGTCR) in the global utilities block to put the device in the doze or sleep states.

Note that the software that writes to either DEVDISR or POWMGTCR can be running either on the e500 core or on an external master that can write to the MPC8560 memory-mapped registers through the RapidIO or PCI interfaces.

These features are described in further detail in this section.

17.5.1.1 Relationship Between Core and Device Power Management States

The MPC8560 has three low-power states: doze, nap, and sleep. The mapping of core and device power management states is shown in Figure 17-19 showing state transitions from the perspective of the e500 core.

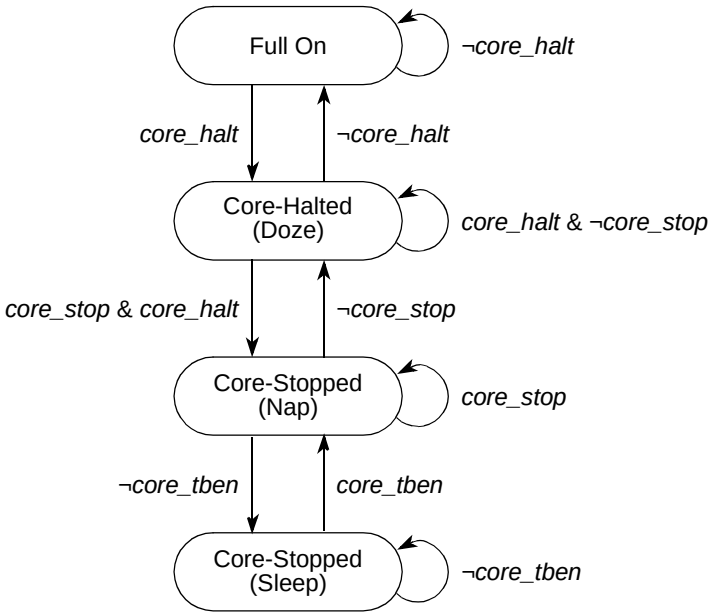


Figure 17-19. e500 Core Power Management State Diagram

For each operating state represented in the diagram, the core's state is listed first, with the corresponding state of the MPC8560 shown beneath it in parenthesis. Note that there are many other variables that control the state transitions between MPC8560 power management states. These additional variables are described in more detail in Section 17.5.1.7, "Power-Down Sequence Coordination."

Table 17-22 lists basic characteristics of the low-power modes and the full on mode.

Table 17-22. MPC8560 Power Management Modes—Basic Description

Mode	Description	Core Responds To		Signal States	
		Snoop	Interrupts	READY	ASLEEP
Full On	All units operating normally	Yes	Yes	Asserted	Negated
Doze	Core stops dispatching new instructions (core is halted)	Yes	Yes	Negated	Negated
Nap	Core is stopped with clocks off except to time base Should flush data cache before entering	No	Yes	Negated	Negated
Sleep	Core is stopped with clocks off. Clocks powered down to all blocks (including core time base) except to the interrupt controller (PIC) unit	No	Yes	Negated	Asserted

17.5.1.2 $\overline{\text{CKSTP\_IN}}$ Is Not Power Management

$\overline{\text{CKSTP\_IN}}$ is not described here because it is not considered a power management signal, although asserting it does stop the core and a stopped core is technically in a low-power mode. $\overline{\text{CKSTP\_IN}}$ is described in Section 17.3.2, "Detailed Signal Descriptions."

17.5.1.3 Dynamic Power Management

Many blocks in the MPC8560 can dynamically turn off clocks within the block when sections of the block are idle. This feature is always enabled and occurs automatically.

17.5.1.4 Shutting Down Unused Blocks

As described in Section 17.4.1.11, "Device Disable Register (DEVDISR)," DEVDISR provides a way to shut down certain functional blocks within the MPC8560 when they are not needed in a particular system. DEVDISR 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.

DEVDISR was designed with the expectation that, once initialized by software, it would be modified only by a hard system reset ($\overline{\text{HRESET}}$). It is recommended that this register be written only during system initialization. Blocks disabled by DEVDISR must not be re-enabled without a hard reset. (Setting DEVDISR[TB] disables the core's timer facilities, and setting DEVDISR[E500] places the core in the core-stopped state in which it does not respond to

interrupts.) The results of re-enabling previously disabled blocks (by clearing the corresponding DEVDISR field) without a hard reset are boundedly undefined.

NOTE

Functional blocks disabled using DEVDISR cannot respond to configuration accesses. Any access to configuration, control, and status registers of a disabled block is a programming error.

17.5.1.5 Software-Controlled Power-Down States

e500 software can place the device in doze, nap, or sleep power-down states by writing to HID0 in the core. In addition, external masters can write to the memory-mapped POWMGTCR in the MPC8560 to cause the device to enter doze or sleep modes.

17.5.1.5.1 Doze Mode

In doze mode, the e500 core suspends instruction execution, significantly reducing the power consumption of the core. Snooping of the L1 data cache is still supported and thus the data in the data cache is kept coherent. Interrupts directed to the core as described in Section 10.1.3, “Interrupts to the Processor Core,” are monitored by the device and cause the MPC8560 to use the defined handshake mechanism to exit the core from doze mode to allow the core to recognize and process the interrupt; however, unless the interrupt subroutine turns off (or masks) the control bits that enabled doze mode (MSR[WE], and HID0[DOZE]), the device re-enters doze mode after the interrupt has been serviced. See Section 17.5.1.8, “Interrupts and Power Management,” for more information.

The e500 core’s timer facilities are still enabled during doze mode, and core time base interrupts can be generated. All device logic external to the core remains fully operational in doze mode.

17.5.1.5.2 Nap Mode

In nap mode all clocks internal to the e500 core are turned off except for its timer facilities clock (the core time base). The L1 caches do not respond to snoops in nap mode, so if coherency with external I/O transactions is required, the L1 cache must be flushed before entering nap mode.

Similar to doze mode, interrupts occurring in nap mode cause the device to wake up the e500 core in order to service the interrupt. However, unless the interrupt service routine changes the control bits that caused the device to enter nap mode (MSR[WE], and HID0[NAP]), the MPC8560 returns to nap mode after the interrupt is serviced. See Section 17.5.1.8, “Interrupts and Power Management,” for more information.

All device logic external to the e500 core remains fully operational in nap mode.

17.5.1.5.3 Sleep Mode

In sleep mode, all clocks internal to the e500 core are turned off, including the timer facilities clock. All I/O interfaces in the device logic are also shut down. Only the clocks to the MPC8560 PIC are still running so that an external interrupt can wake up the device. Note that the DDR controller does not shut down unless DDR_SDRAM_INTERVAL[REFINT] is set to a non-zero value. See Section 9.4.1.7, "DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)," for details. Note that external interrupts from port C of the CPM are a special case and do not reach the PIC when the device is asleep. Therefore, they do not cause the device to wake up.

After the core and I/O interfaces have shut down, ASLEEP is asserted and READY is negated.

NOTE

Only external interrupts can wake the MPC8560 from sleep mode. Internal interrupt sources like the core interval timer or watchdog timer depend on an active clock for their operation and these are disabled in sleep mode.

17.5.1.6 Power Management Control Fields

The e500 core provides the following fields to signal power management requests to the MPC8560 device logic.

- MSR[WE]—Used to qualify the values of HID0[DOZE,NAP,SLEEP] in the generation of the internal *doze*, *nap*, and *sleep* signals
- HID0[DOZE]—Signals the MPC8560 to initiate doze mode
- HID0[NAP]—Signals the MPC8560 to initiate nap mode
- HID0[SLEEP]—Signals the MPC8560 to initiate sleep mode

These register fields and their functional relationship are shown in Figure 17-20. The *e500 Reference Manual* has details on accessing these power management control bits.

An external master can also initiate power management requests by setting the DOZ or SLP bits in the memory-mapped power management control and status register (POWMGTCSR). Because the core responds to snoops while dozing but not while napping, maintaining cache coherency requires significant preparation by the core before entering nap mode. For this reason only the core can initiate a nap during normal operation while other masters can initiate a doze.

17.5.1.7 Power-Down Sequence Coordination

To preserve cache coherency and otherwise avoid loss of system state, the core's transition to low-power modes is coordinated by a set of handshaking signals, shown in Figure 17-20, and protocols with all other MPC8560 functional blocks that respond to power-down requests. The

mode-transition protocol is executes automatically under these conditions and is shown in Figure 17-19 and described in Table 17-23.

The column in Table 17-23 showing the global utilities block as initiating a low-power mode corresponds to the external masters that can write to the POWMGTCSR that resides in the global utilities block. For the MPC8560, these are the RapidIO and PCI/PCI-X interfaces. However, note that the core can also write to POWMGTCSR and, in this case, can initiate power management through the global utilities block.

Table 17-23. Power Management Entry Protocol and Initiating Functional Units

Low-Power Mode	Entry Protocol	Initiating Functional Unit	
		Global Utilities	Core
Doze	1. Assert *core_halt* input to core. 2. Wait for *core_halted* handshake from core.	√	√
Nap	1. Follow doze protocol 2. Assert *core_stop* input to core. 3. Wait for *core_stopped* handshake from core.	—	√
Sleep	1. Follow doze protocol; send stop requests to rest of device. 2. Follow nap protocol. 3. Wait for all interfaces to acknowledge stop requests. 4. Assert ASLEEP, negate READY, power down all clocks except to PIC unit	√	√

As shown in Figure 17-20, the e500 core enters low-power modes only in response to the *core_halt*, *core_stop*, or *core_tben* inputs from the MPC8560's power management logic. These inputs may be prompted by the core (by setting the NAP, DOZE, or SLEEP bits in the HID0 when enabled by setting MSR[WE]) or by an external master (by setting POWMGTCSR[DOZ,SLP].

Figure 17-20 shows how all the clocking to the core timer facilities is disabled by clearing HID0[TBEN]. When enabled, (HID0[TBEN] = 1), the clock source is either the CCB clock divided by eight (the default) or a synchronized version of the RTC input. For more details, see Section 6.10.1, "Hardware Implementation-Dependent Register 0 (HID0)."

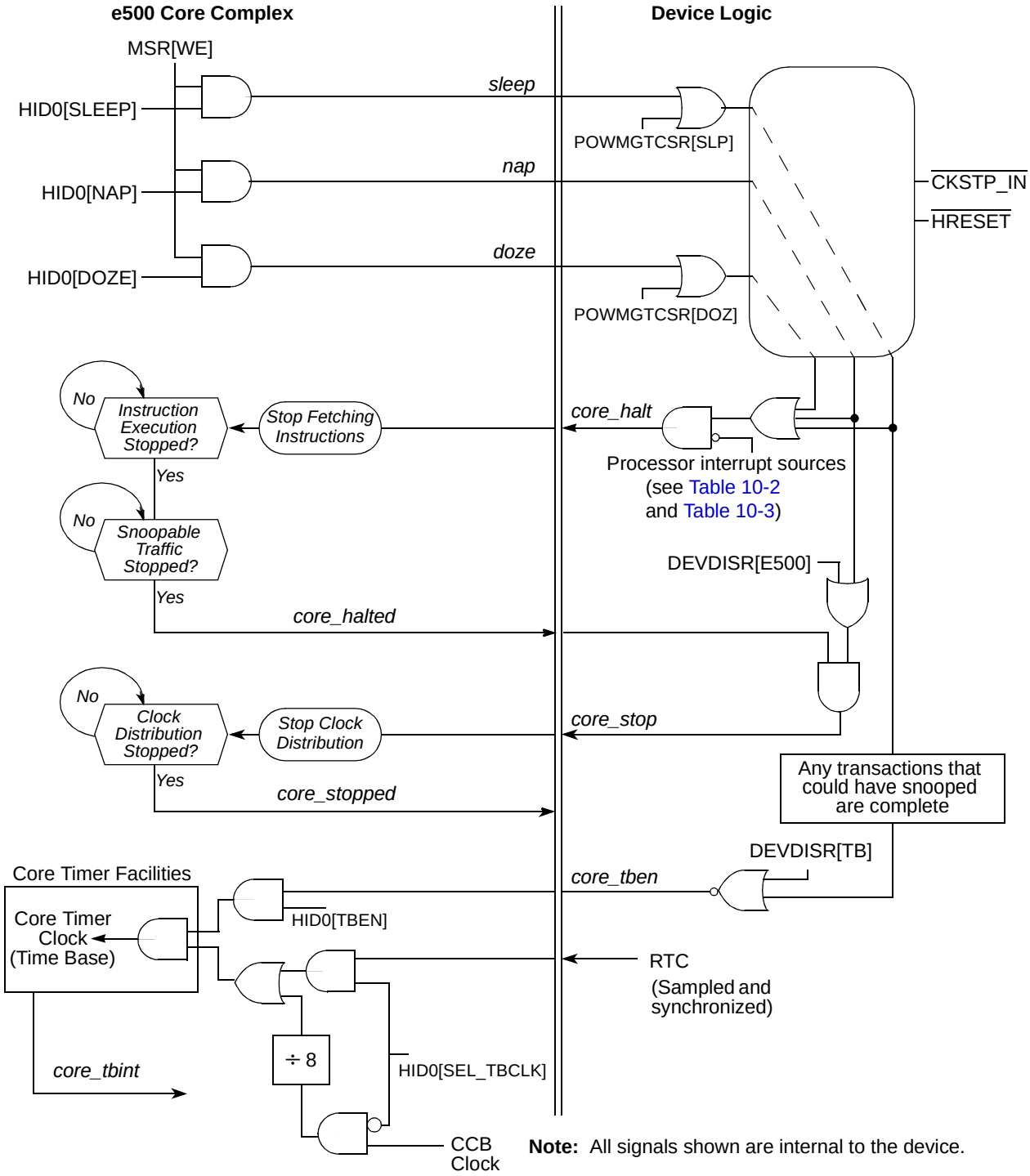


Figure 17-20. MPC8560 Power Management Handshaking Signals

17.5.1.8 Interrupts and Power Management

Whether low-power modes are automatically re-enabled after an interrupt is processed differs depending on whether the low power mode was entered due to a write to the core MSR[WE] bit or the low power mode was entered due to a write to POWMGTCSR.

17.5.1.8.1 Interrupts and Power Management Controlled by MSR[WE]

When an interrupt is asserted to the CPU, the core complex saves portions of the MSR to MCSRR1, CSRR1, or SRR1 (depending on the type of interrupt), and restores those values on return from the routine. MSR[WE], which gates the *doze*, *nap*, and *sleep* power management outputs (internal device signals) from the core complex, is always among the bits saved and restored; hence these outputs negate to the MPC8560 power management logic when the interrupt begins processing in the core. They return to their previous state when the core executes an **rfi**, **rfci**, or **rfmci** instruction. Section 10.1.3, "Interrupts to the Processor Core," lists interrupts that cause the MPC8560 to wake up.

NOTE

Returning *doze*, *nap*, and *sleep* signals to their original state when MSR[WE] is restored differs from how power management is implemented on earlier PowerPC devices where MSR[POW], which enables power-down requests, is cleared when the processor exits a low-power state and is not automatically restored, as it is in Book E implementations.

17.5.1.8.2 Interrupts and Power Management Controlled by POWMGTCSR

The IRQ_MSK and CI_MSK fields of the POWMGTCSR register prevent *int* interrupts or *cint* critical interrupts from waking the device from a low power state. This is true regardless of the method used to enter the low power state.

Any unmasked interrupt (not masked by the mask bits in the POWMGTCSR register) causes the POWMGTCSR[DOZ,SLP] fields to be cleared when it occurs. When such an interrupt occurs, the device returns to the normal operating mode and does not automatically attempt to return to a low power state after the interrupt is handled.

Note that interrupts caused by the unconditional debug event ($\overline{\text{UDE}}$) and machine check ($\overline{\text{MCP}}$) signals are not masked by the IRQ_MSK and CI_MSK fields; therefore, when these signals assert, the POWMGTCSR[DOZ,SLP] fields are cleared and the device will return to full power operation. See Section 17.4.1.12, "Power Management Control and Status Register (POWMGTCSR)," for detailed information about the bits of POWMGTCSR.

Note also that unmasked interrupts that occur while the device is in the process of going into the sleep state (before sleep is completely attained) can also cause the device to clear the POWMGTCSR[DOZ,SLP] fields and return the device to full power operation.

17.5.1.9 Snooping in Power-Down Modes

When the MPC8560 is in doze mode, the e500 core is in the core-halted state and it snoops its L1 caches and full coherency is maintained. In deeper power-down modes, however, the e500 core does not respond to snoops.

The MPC8560 does not perform dynamic bus snooping as described in the *e500 Reference Manual*. That is, when the e500 core is in the core-stopped state (which is the state of the core when the MPC8560 is in either the nap or sleep state), the core is not awakened to perform snoops on global transactions. Therefore, before entering nap or sleep modes, the L1 caches should be flushed if coherency is required during these power-down modes.

17.5.1.10 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. It is assumed that the desired power-saving state (doze, nap, or sleep) was set up by setting the appropriate HID0 bit, typically at system start-up time. Setting WE has no direct effect on instruction execution, but is reflected on the internal *doze*, *nap*, and *sleep* signals, depending on the HID0 settings. To ensure a clean transition into and out of a power-saving mode, the following program sequence is recommended:

```
        sync
        mtmsr (WE)
        isync
loop:   br loop
```

17.5.1.11 Requirements for Reaching and Recovering from Sleep State

In order to successfully reach the sleep state, I/O traffic to the device must be stopped. The logic that controls the power down sequence waits for all I/O interfaces to become idle. In some applications this may happen eventually without actively shutting down interfaces, but most likely, software will have to take steps to shut down the TSEC, CPM, PCI, and RapidIO interfaces before issuing the command (either the write to the core MSR[WE] as described above or writing to POWMGTCSR) to put the device into sleep state.

The RapidIO and PCI interfaces will begin retrying inbound transactions before entering a power down state. Upon exiting sleep, the RapidIO interface begins its training sequence to re-establish the link. The PCI interface, however, could potentially be in an unknown state when it exits sleep if it was in the middle of a retry sequence when its internal clocks were shut down. Therefore it is strongly recommended that system software clear the memory space bit in the PCI Bus Command Register before putting the device in sleep mode. Software may also need to set the Agent Config

Lock bit of the PCI Bus Function Register so that the device will not respond to configuration transactions. Upon exiting sleep mode, software should return these configuration bits to their normal state.

17.5.2 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 are the following:

- PCI_AD[47:40] and PCI_AD[39:32]. When configured as general-purpose I/O, PCI_AD[47:40] function as outputs and PCI_AD[39:32] function as inputs.

  PCI_AD[47:32] 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.
- TSEC2_RxD[0:7] and TSEC2_TxD[0:7]. 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 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.

When configured as general-purpose I/O signals, software can read inputs by reading the associated GPIO data register. Output values can be set by writing the to the associated GPIO data register. For details regarding the control and status of the general-purpose I/O signals, see Section 17.4.1.7, "General-Purpose I/O Control Register (GPIOCR)."

17.5.3 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:

- $\overline{\text{LCS}}$[5:7] are multiplexed with DMA channel 2 $\overline{\text{DMA\_DREQ2}}$, $\overline{\text{DMA\_DACK2}}$, and $\overline{\text{DMA\_DONE2}}$.
- IRQ[9:11] are multiplexed with DMA channel 3 $\overline{\text{DMA\_DREQ3}}$, $\overline{\text{DMA\_DACK3}}$, and $\overline{\text{DMA\_DDONE3}}$.

For details regarding the selection of the alternate function DMA trigger, see Section 17.4.1.10, "Alternate Function Signal Multiplex Control Register (PMUXCR)."

The multiplexing of the CPM signals occurs through the CPM programming model. See Chapter 45, “Parallel I/O Ports,” for details on CPM signal multiplexing.

Chapter 18
Performance Monitor

This chapter describes the MPC8560 performance monitor facility, which can be used to monitor and optimize performance. The e500 core implements a separate performance monitor for strictly core-related behavior, such as instruction timing and L1 cache operations. This is described in the *e500 Reference Manual*.

Section 18.4.7, "Performance Monitor Events," briefly describes the events that can be monitored. Refer to the individual chapters for a better understanding of these events.

18.1 Introduction

The MPC8560 includes a performance monitor facility that can be used to monitor and record selected behaviors of the integrated device. Although the performance monitor described here is similar in many respects to the performance monitor facility implemented on the e500 core, it differs in that it is implemented using memory-mapped registers and it counts events outside the e500 core, for example, RapidIO, PCI, DDR, and L2 cache events.

Performance monitor counters (PMC0–PMC8) are used to count events selected by the performance monitor local control registers. PMC0 is a 64-bit counter specifically designated to count cycles. PMC1–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 is a high-level block diagram of the performance monitor, which 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

counters and takes priority over local control registers. The local control registers are divided into two groups as follows:

- Local control A registers control counter freezing, overflow condition enable, event selection, and burstiness. Local control register PMLCA0, which controls counter PMC0, does not contain event selection because PMC0 counts only cycles.
- Local control B registers control the start and stop triggering, contain the counters' threshold values, and the value of the threshold multiplier. Local control register PMLCB0, which controls PMC0, does not contain threshold information because PMC0 only counts cycles.

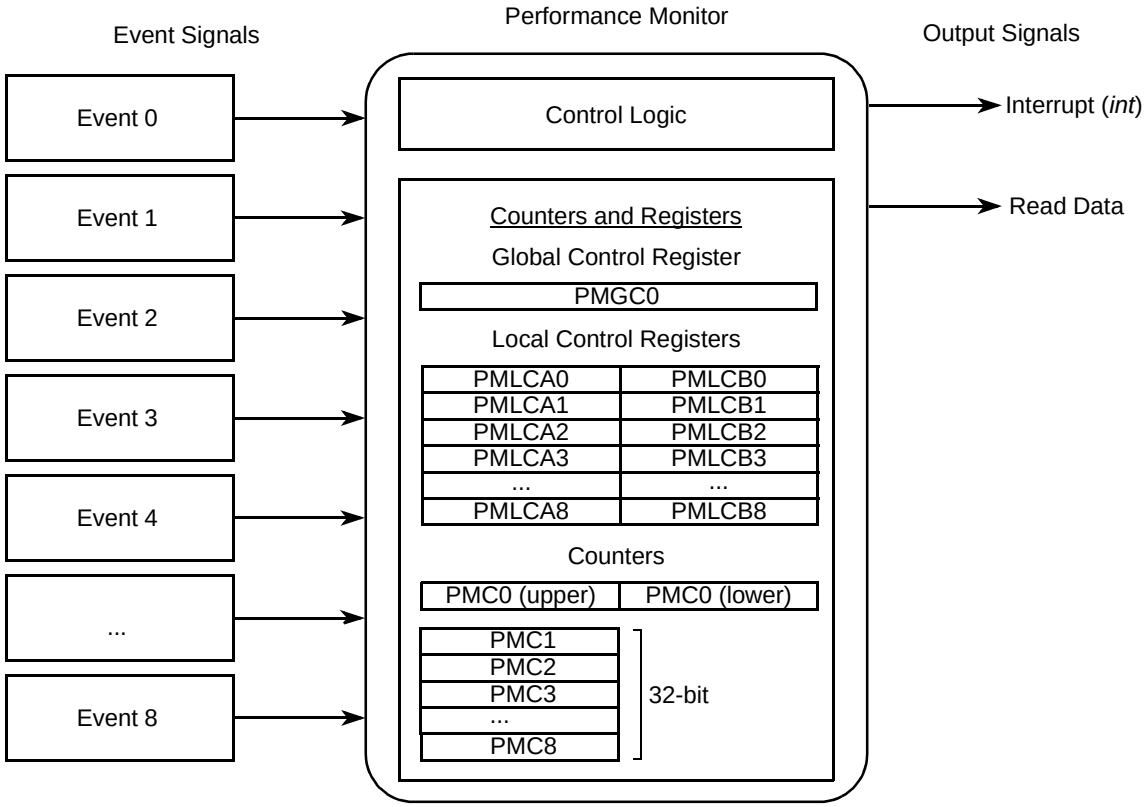


Figure 18-1. Performance Monitor Block Diagram

NOTE

Note that the performance monitor implemented on the core also uses registers named PMC*n* and PMLC*xn*; however, these registers are implemented on-chip, they are accessed using the EIS-defined **mtpmr** and **mtpmr** instructions, and they do not interact with the memory-mapped registers shown in Figure 18-1.

Performance monitor events are signalled by the functional blocks in the integrated device and are selectively recorded in the PMCs. Sixty-four of these events are referred to as reference events,

which can be counted on any of the eight counters. Counter-specific events can be counted only on the counter where the event is defined.

The performance monitor can generate an interrupt on overflow. Several control registers specify how a performance monitor interrupt is signalled. The PMCs can also be programmed to freeze when an interrupt is signalled.

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 Signal Descriptions

The performance monitor does not have any signals that are driven externally (off-chip) but it does assert the internal interrupt (*int*) signal on a performance monitor interrupt condition.

18.3 Memory Map and Register Definition

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 18-1 lists the performance monitor registers. These registers can be read or written only 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. Two local control registers are associated with each counter in addition to a global control register that applies to all counters.

Table 18-1. Control Register Memory Map

Address Offset (in Hex)	Register	Access	Reset	Section/Page
0xE_1000	PMGC0—Performance monitor global control register	R/W	0x0000_0000	18.3.2.1/18-5
0xE_1010	PMLCA0—Performance monitor local control register A0	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1014	PMLCB0—Performance monitor local control register B0	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1018	PMC0 (upper)—Performance monitor counter 0 upper	R/W	0x0000_0000	18.3.3.1/18-9
0xE_101C	PMC0 (lower)—Performance monitor counter 0 lower	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1020	PMLCA1—Performance monitor local control register A1	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1024	PMLCB1—Performance monitor local control register B1	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1028	PMC1—Performance monitor counter 1	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1030	PMLCA2—Performance monitor local control register A2	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1034	PMLCB2—Performance monitor local control register B 2	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1038	PMC2—Performance monitor counter 2	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1040	PMLCA3—Performance monitor local control register A3	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1044	PMLCB3—Performance monitor local control register B3	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1048	PMC3—Performance monitor counter 3	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1050	PMLCA4—Performance monitor local control register A4	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1054	PMLCB4—Performance monitor local control register B4	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1058	PMC4—Performance monitor counter 4	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1060	PMLCA5—Performance monitor local control register A5	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1064	PMLCB5—Performance monitor local control register B 5	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1068	PMC5—Performance monitor counter 5	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1070	PMLCA6—Performance monitor local control register A6	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1074	PMLCB6—Performance monitor local control register B6	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1078	PMC6—Performance monitor counter 6	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1080	PMLCA7—Performance monitor local control register A7	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1084	PMLCB7—Performance monitor local control register B7	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1088	PMC7—Performance monitor counter 7	R/W	0x0000_0000	18.3.3.1/18-9
0xE_1090	PMLCA8—Performance monitor local control register A8	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1094	PMLCB8—Performance monitor local control register B8	R/W	0x0000_0000	18.3.2.2/18-5
0xE_1098	PMC8—Performance monitor counter 8	R/W	0x0000_0000	18.3.3.1/18-9

In addition to these registers, the interrupt control provides four pairs of mask registers that can be used to monitor message, interprocessor, timer, and external interrupts. See Section 10.3.4, “Performance Monitor Mask Registers (PMMRs).”

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 PMCs.

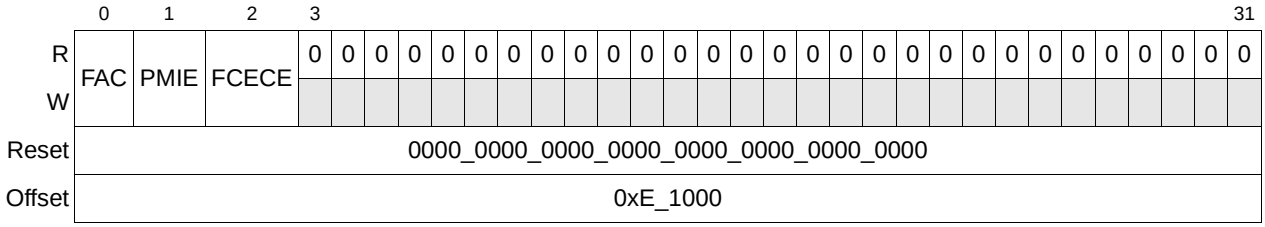


Figure 18-2. Performance Monitor Global Control Register (PMGC0)

Table 18-2 describes PMGC0 fields.

Table 18-2. PMGC0 Field Descriptions

Bits	Name	Description
0	FAC	Freeze all counters. 0 PMCs are incremented (if permitted by other PMGC0/PMLC bits). 1 PMCs are not incremented. Set by hardware when an interrupt is signalled and FCECE =1
1	PMIE	Performance monitor interrupt enable. Interrupts are caused by PMC overflows. 0 Interrupts are disabled. 1 Interrupts are enabled and occur when an enabled condition or event occurs.
2	FCECE (DISCOUNT)	Freeze counters on enabled condition or event. An enabled condition or event is defined as: The msb = 1 in PMC*n* and PMLCA*n*[CE] = 1. The use of the trigger and freeze counter conditions depends on the enabled condition. 0 PMCs can be incremented (if permitted by other control bits). 1 PMCs can be incremented (if permitted by other control bits) only until an enabled condition or event occurs, at which time PMGC0[FAC] is set. It is up to software to clear FAC.
3–31	—	Reserved

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 PMCs. The performance monitor local control A and B registers are paired 32-bit control registers that are associated with an individual counter to specify how the counter is used and what event is monitored on that counter. Figure 18-3 shows the performance monitor local control A0 register (PMLCA0).

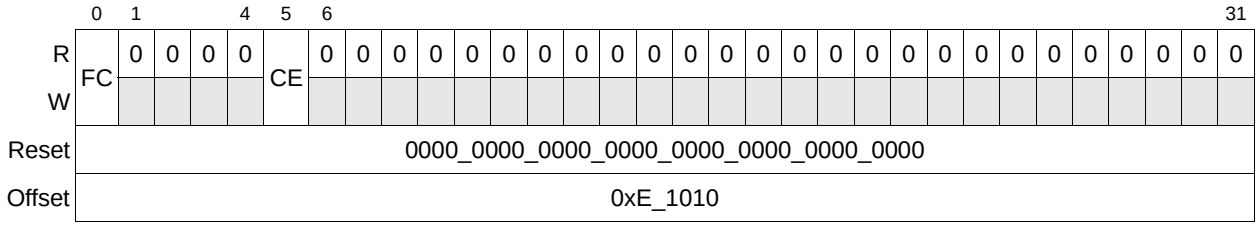


Figure 18-3. Performance Monitor Local Control Register A0 (PMLCA0)

Table 18-3 describes PMLCA0 fields.

Table 18-3. PMLCA0 Field Descriptions

Bits	Name	Description
0	FC	Freeze counter. Basic counter enable 0 The PMCs are enabled and incremented (if permitted by other SPM control bits). 1 The PMCs are disabled–they do not increment.
1–4	—	Reserved
5	CE	Condition enable. Controls counter overflow condition. Should be cleared when PMC0 is used as a trigger or is selected for chaining 0 Overflow conditions for PMC0 cannot occur (PMC0 cannot cause interrupts or freeze counters) 1 Overflow conditions occur when PMC0[msb] is set.
6–31	—	Reserved

Figure 18-4 shows the performance monitor local control registers A1–A8.

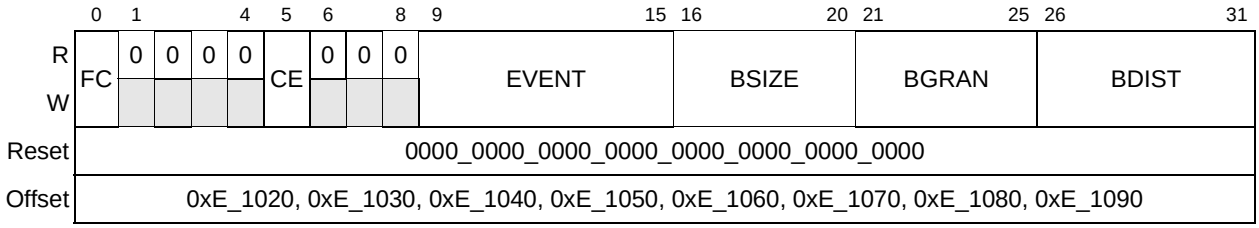


Figure 18-4. Performance Monitor Local Control Registers A1–A8 (PMLCA1–PMLCA8)

Table 18-4 describes PMLCA*n* fields.

Table 18-4. PMLCA1–PMLC8 Field Descriptions

Bits	Name	Description
0	FC	Freeze counter 0 The PMCs are incremented (if permitted by other PMC control bits). 1 The PMCs are not incremented (if permitted by other PMC control bits).
1–4	—	Reserved

Table 18-4. PMLCA1–PMLC8 Field Descriptions (continued)

Bits	Name	Description
5	CE	Condition enable 0 Overflow conditions for PMC*n* cannot occur (PMC*n* cannot cause interrupts or freeze counters). Should be cleared when PMC*n* is used as a trigger or is selected for chaining. 1 Overflow conditions occur when PMC*n*[msb] is set.
6–8	—	Reserved
9–15	EVENT	Event selector. Up to 128 events selectable. See Table 18-10 for definition of events.
16–20	BSIZE	Burst size. Fewest event occurrences that constitute a burst, that is, a rapid sequence of events followed by a relatively long pause. A value less than two implies regular event counting. Any non-threshold, regular event may be counted in a bursty fashion. See Section 18.4.6, “Burstiness Counting,” for more information.
21–25	BGRAN	Burst granularity. The maximum number of clock cycles between events that are considered part of a single burst. See Section 18.4.6, “Burstiness Counting.”
26–31	BDIST	Burst distance (used with TBMULT). The number of clock cycles between bursts. Must be set to a value greater than BSIZE for proper burstiness counting behavior. 00_0000 Regular counting

Figure 18-5 shows the performance monitor local control B0 register (PMLCB0).

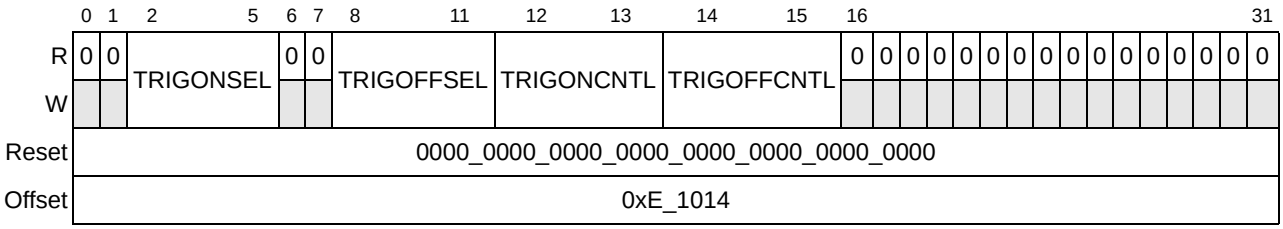


Figure 18-5. Performance Monitor Local Control Register B0 (PMLCB0)

Table 18-5 describes PMLCB0 fields.

Table 18-5. PMLCB0 Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–5	TRIGONSEL	Trigger-on select. The number of the counter that starts event counting. When the specified counter’s TRIGONCNTL event overflows, the current counter begins counting. No triggering occurs if the value is self-referential, that is, when set to the current counter number.
6–7	—	Reserved
8–11	TRIGOFFSEL	Trigger-off select. The number of the counter that stops event counting. When the specified counter’s TRIGONCNTL event overflows, the current counter stops counting. No triggering occurs if the value is self-referential, that is, when set to the current counter number.

Table 18-5. PMLCB0 Field Descriptions (continued)

Bits	Name	Description
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 performance monitor local control registers B1–B8.

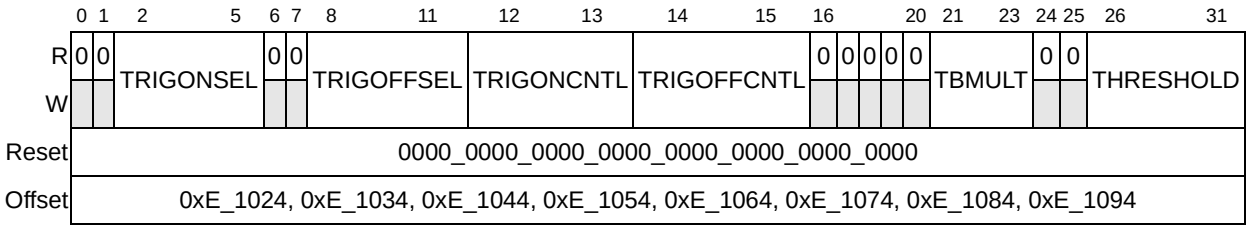


Figure 18-6. Performance Monitor Local Control Registers B1–B8 (PMLCB1–PMLCB8)

Table 18-5 describes PMLCB*n* fields.

Table 18-6. PMLCB*n* Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2–5	TRIGONSEL	Trigger-on select. Set this field equal to the number of the counter that should trigger event counting to start. When the specified counter's TRIGONCNTL event overflows, the current counter begins counting. No triggering occurs when TRIGONSEL = current counter.
6–7	—	Reserved
8–11	TRIGOFFSEL	Trigger-off select. Set this field equal to the number of the counter that should trigger event counting to stop. When the specified counter's TRIGONCNTL event overflows, the current counter stops counting. No triggering occurs when TRIGOFFSEL = current counter.
12–13	TRIGONCNTL	Trigger-on control. Indicates the condition under which triggering to start counting occurs 00 Trigger off (no triggering to start) 01 Trigger on change 10 Trigger on overflow 11 Reserved

Table 18-6. PMLCB*n* Field Descriptions (continued)

Bits	Name	Description
14–15	TRIGOFFCNTL	Trigger-off control. Indicates the condition under which triggering to stop occurs 00 Trigger off (no triggering to stop) 01 Trigger on change 10 Trigger on overflow 11 Reserved
16–20	—	Reserved
21–23	TBMULT	Threshold and burstiness multiplier. Threshold events are counted when the event duration exceeds a specified threshold value. The threshold is scaled based on the TBMULT settings. The burst distance for burstiness counting is also scaled using the TBMULT settings. For all events that scale the threshold, the threshold field is multiplied by the factors shown below (ranging from 1 to 128). 000 1 001 2 010 4 011 8 100 16 101 32 110 64 111 128
24–25	—	Reserved
26–31	THRESHOLD	Threshold. Only events whose (number of) occurrences exceed this value are counted. By varying the threshold value, software can characterize the events subject to the threshold. For example, if PMC2 counts TSEC BD read latencies for which the duration exceeds the threshold, software can obtain the distribution of TSEC BD read latencies 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 PMCs in detail.

NOTE

Because accessing a PMC manually has priority over incrementing it due to event counting, reading or writing a PMC while it is counting may affect the count. Likewise, accessing a performance monitor control register while its target counter is counting may also affect the count.

18.3.3.1 Performance Monitor Counters (PMC0–PMC8)

PMC0–PMC8 are used to count events selected by the performance monitor local control registers. PMC0, shown in Figure 18-7, is associated with two 32-bit registers that form a 64-bit counter designated to count clock cycles. PMC0 upper represents the upper 32 bits of counter 0, and PMC0 lower represents the lower 32 bits.

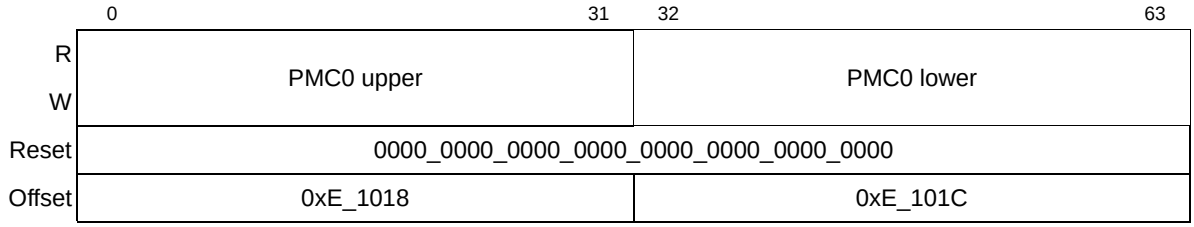


Figure 18-7. Performance Monitor Counter Register 0 (PMC0)

Table 18-7 describes PMC0 fields.

Table 18-7. PMC0 Field Descriptions

Bits	Name	Description
0–63	PMC0	Event count. Counts only clock cycles

PMC1–PMC8, shown in Figure 18-8, are 32-bit counters that can monitor 64 unique events in addition to the 64 reference events that can be counted on all of these registers.

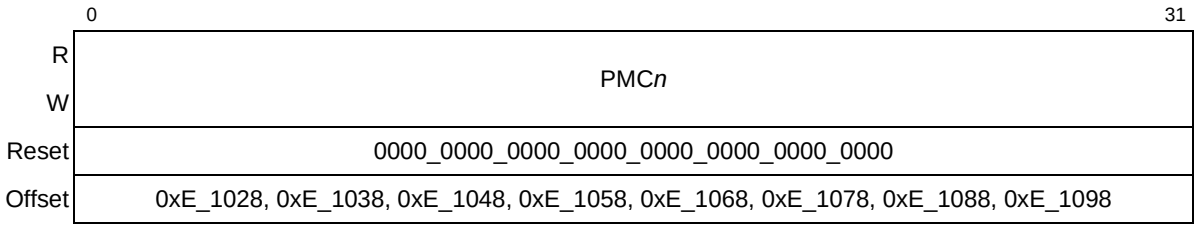


Figure 18-8. Performance Monitor Counter Registers 1–8 (PMC1–PMC8)

Table 18-8 describes PMC*n* fields.

Table 18-8. PMC*n* Field Descriptions

Bits	Name	Description
0–31	PMC*n*	Event count. An overflow is indicated when the msb = 1. Manually setting the msb can cause an immediate interrupt.

18.4 Functional Description

This section describes the use of some features of the performance monitor.

18.4.1 Performance Monitor Interrupt

PMCs can generate an interrupt on an overflow when the msb of a counter changes from 0 to 1. For the interrupt to be signalled, the condition enable bit (PMLCA*n*[CE]) and performance monitor interrupt enable bit (PMGC0[PMIE]) must be set. When an interrupt is signalled and the

freeze-counters-on-enabled-condition-or-event bit (PMGC0[FCECE]) is set, PMGC0[FAC] is set by hardware and all of the registers are frozen. Software can clear the interrupt condition by resetting the performance monitor and clearing the most significant bit of the counter that generated the overflow.

18.4.2 Event Counting

Using the control registers described in Section 18.3.2, “Control Registers,” the nine PMCs can count the occurrences of specific events. The 64-bit PMC0 is designated to count only clock cycles. However, to provide flexibility, a total of 64 reference events can be counted on any of the 32-bit PMCs (PMC1–PMC8). Additionally, up to 64 unique events can be counted on each 32-bit counter.

The performance monitor must be reset before event counting sequences. The performance monitor can be reset by first freezing one or more counters and then clearing the freeze condition to allow the counters to count according to the settings in the performance monitor registers.þCounters can be frozen individually by setting PMLCAn[FC] bits, or simultaneously by setting PMGC0[FAC]. Simply clearing these freeze bits will then allow the performance monitor to begin counting based on the register settings.þ

Note that using PMLCAn[FC] to reset the performance monitor resets only the specified counter. Performance monitor registers can be configured through reads or writes while the counters are frozen as long as freeze bits are not cleared by the register accesses.

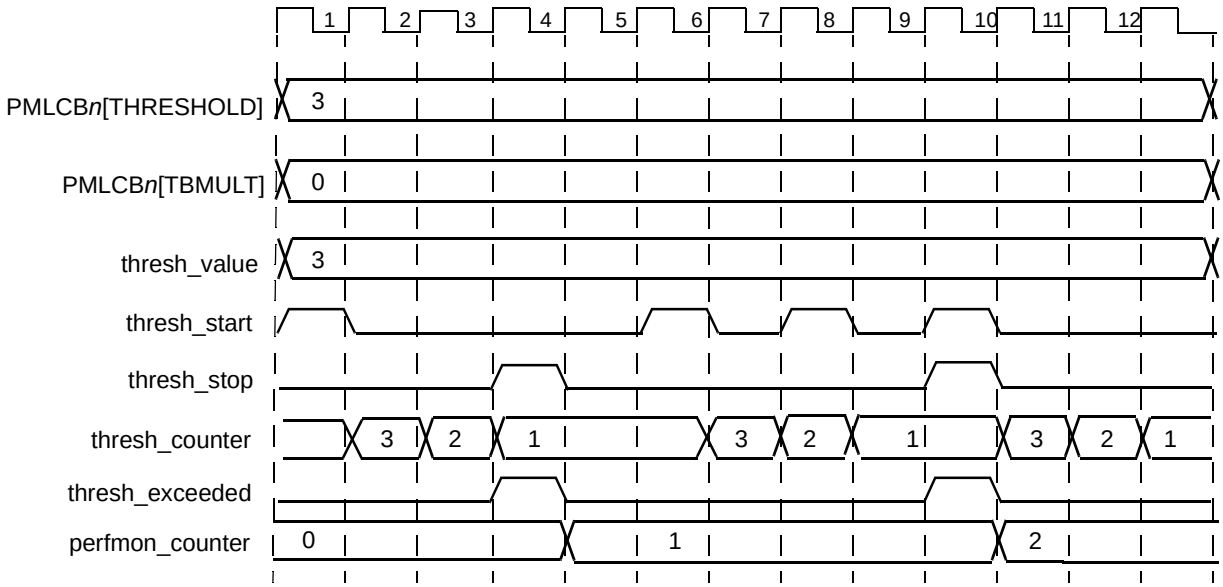
18.4.3 Threshold Events

The threshold feature allows characterization of events that can take a variable number of clock cycles to occur. Threshold events are counted only if the latency is greater than the threshold value specified in PMLCB*n*[THRESHOLD].

For duration threshold event sequences, the PMC increments only when the duration of the event is equal to or greater than the threshold value. The threshold value is scaled by a multiple specified in PMLCB*n*[TBMULT].

A threshold event requires two signals: The first indicates when a threshold event sequence begins, and the second indicates when it ends. An internal counter determines when the threshold count is exceeded and when the PMC can increment. This internal counter decrements during a threshold event sequence until it reaches the value of one. A new sequence cannot begin until the current one completes. Additional threshold start signals are ignored during a sequence until a threshold stop signal occurs. If both a start and stop signal are asserted during the same cycle in a current sequence, the stop terminates the current sequence and the start signals the beginning of a new one. However, if both signals are asserted during the same cycle while not in a current event sequence, both signals are ignored. Figure 18-9 is a timing diagram for duration threshold event counting.

An illegal condition exists if the threshold value obtained from PMLCB*n*[THRESHOLD] and PMLCB*n*[TBMULT] is less than two. Under these conditions the intent of threshold counting is ambiguous.



1 For this example a threshold value of three indicates that the user wishes to count the number of times a particular event lasts three cycles or longer.

Figure 18-9. 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 available chaining events, as shown in Table 18-10, are dedicated to specific registers.

Note that the chaining overflow event occurs when the counter reaches its maximum value and wraps, not when the register's msb is set. For this overflow to occur, PMLCA*n*[CE] should be cleared to avoid signalling an interrupt when the counter's most-significant bit is set. Note that several cycles may be required for the chained counters to reflect the true count because of the internal delay between when an overflow occurs and a counter increments.

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 done by appropriately setting the trigger-on select bit or trigger-off select bit (PMLCB1[TRIGOFFSEL] or PMLCB1[TRIGONSEL]). Additionally, the condition that triggers the counter must be selected by configuring the corresponding control bits (PMLCB1[TRIGONCNTL] or PMLCB1[TRIGOFFCNTL]). Assuming the counter is enabled by other control register settings, the counter increments (or freezes) when its specified event occurs after the trigger-on (or off) condition occurs.

When trigger on and trigger off are both selected, the trigger-off condition is ignored until the trigger-on condition has occurred. Furthermore, when a trigger-off condition occurs, the counter state is preserved; it is not restarted by subsequent trigger-on conditions.

Triggering is disabled when the counter's trigger-select bits specify itself as the trigger source. Similarly, triggering is disabled when the trigger control bits are cleared.

18.4.6 Burstiness Counting

The burstiness counting feature makes it easier to characterize events that occur in rapid succession followed by a relatively long pause. As shown in Table 18-9, event bursts are defined by size, granularity, and distance.

Table 18-9. Burst Definition

Parameter	Description	Register Field
Size	The minimum number of events constituting a burst	PMLCA*n* [BSIZE]
Granularity	The maximum time between individual events counted as members of the same burst	PMLCA*n* [BGRAN]
Distance	The minimum time between bursts	PMLCA*n* [BDIST] x PMLCB*n*[TBMULT]

Figure 18-10 shows the relationships between size, granularity, and distance. Burstiness counting can be performed for all events except threshold events.

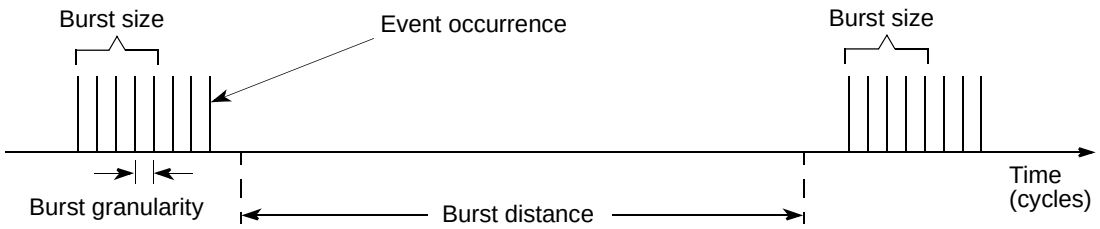


Figure 18-10. Burst Size, Distance, Granularity, and Burstiness Counting

The burstiness size field (PMLCA*n*[BSIZE]) specifies the minimum number of event occurrences that constitute a burst. A burst is identified when the number of event occurrences equals or exceeds PMLCA*n*[BSIZE]. Furthermore, these individual event occurrences must be separated by no more clock cycles than the value in the burstiness granularity field (PMLCA*n*[BGRAN]). Note

that, although a burst is identified when the minimum number of events occurs, it is not counted until the burst sequence has ended. A burst sequence ends when the specified burstiness granularity is exceeded, at which point the last valid event has occurred for that sequence.

PMLCA*n*[BGRAN] specifies the maximum number of cycles between individual events for them to qualify as members of the same burst sequence.

The burstiness distance field (PMLCA*n*[BDIST]) and threshold/burstiness multiplier field (PMLCB*n*[TBMULT]) specify the acceptable number of cycles between the end of a burst sequence and the beginning of a new sequence for a group of event occurrences to be counted as an individual burst. The product of the burstiness distance field and the threshold/burstiness multiplier field determine the burstiness distance value used to determine when another burst sequence can begin. Note that the burst distance count begins when a new burst sequence ends and the PMC is incremented. No new burst sequence may begin until the burst distance count has reached zero. After the burst distance count reaches zero, it holds the zero value indicating that a new burst sequence can be counted. The burst distance count begins again when a new burst sequence is identified and counted.

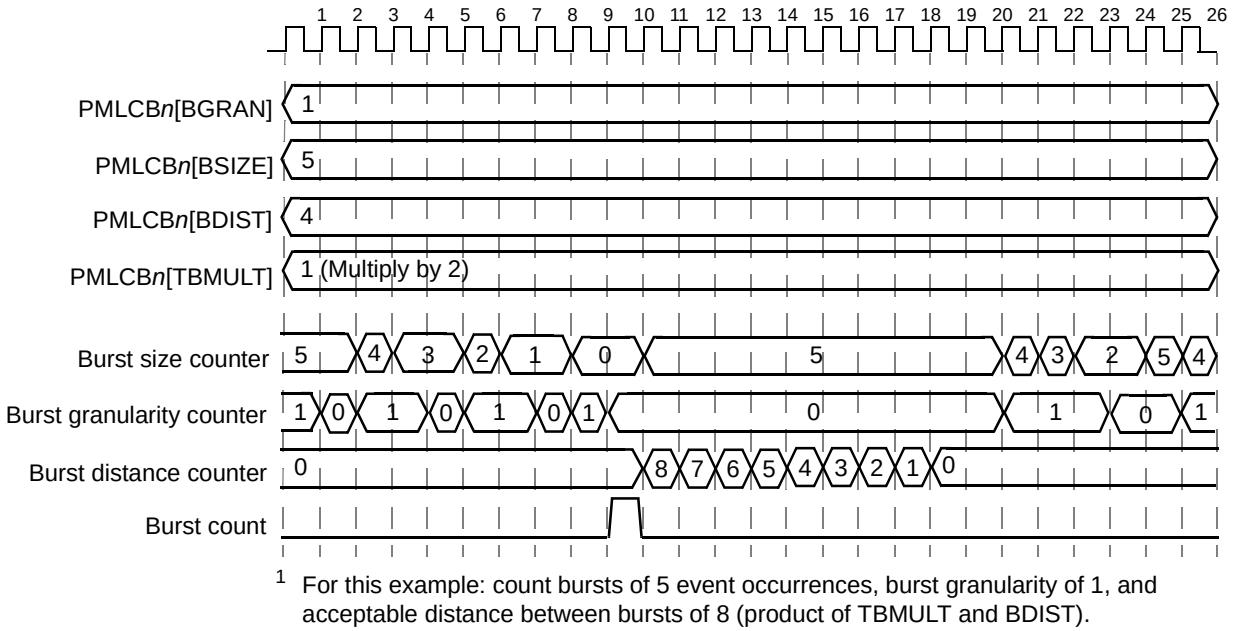
Burstiness counting is disabled when the definition of a burst is ambiguous, that is, when the burst size field is less than two, or the burst distance is zero. When burstiness counting is disabled, regular counting is allowed.

Figure 18-10 shows that the burst distance is measured from the end of one burst sequence and that a new burst sequence may not begin until the burst distance count expires.

Three internal counters track the different values required for burstiness counting.

- Burstiness size is monitored by a counter. It is loaded with the value specified in the local control register when the burst granularity counter and the burst distance counters reach zero, and no new event is occurring. It always decrements when the following conditions occur: its value is not already zero, an event occurs, and the burst distance count equals zero.
- Burstiness granularity is monitored by a counter that is loaded with the specified value in the local control register on the rising edge of an event occurrence whenever the burst distance count equals zero. The granularity counter is decremented (if it has not already reached zero) when an event is not occurring and burst distance count equals zero.
- Burstiness distance is measured by a counter that is loaded with the product of PMLCB*n*[BDIST] and PMLCB*n*[TBMULT] when a burst sequence has been identified and counted. This counter is decremented when burstiness counting is enabled (and the counter has not already reached zero).

A burst is counted at the end of a burst sequence when the three burst parameter counters are all equal to zero. Figure 18-11 shows a burstiness counting example.



1 For this example: count bursts of 5 event occurrences, burst granularity of 1, and acceptable distance between bursts of 8 (product of TBMULT and BDIST).

Figure 18-11. Burstiness Counting Timing Diagram

18.4.7 Performance Monitor Events

Table 18-10 lists performance monitor events specified in PMLCA1–PMLC8.

The event assignment column indicates the event's type and number, using the following formats:

- Ref:#—Reference events are shared across counters PMC1–PMC8. The number indicates the event. For example, Ref:6 means that PMC1–PMC8 share reference event 6.
- C[0–8]:#—Counter-specific events. C8 indicates an event assigned to PMC8. Thus C8:62 means PMC8 is assigned event 62 (PIC interrupt wait cycles).

  Note that with counter-specific events, an offset of 64 must be used when programming the field, because counter-specific events occupy the bottom 64 values of the 7-bit event field where events are numbered. For example, to specify counter-specific event 0, the event field must be programmed to 64.

Counter events not specified in Table 18-10 are reserved.

Table 18-10. Performance Monitor Events

Event Counted	Number	Description of Event Counted
General Events		
Nothing	Ref:0	Register counter holds current value
System cycles	C0	CCB (platform) clock cycles

Table 18-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
DDR Memory Controller Events		
Cycles a read is returning data from DDR SDRAM	Ref:10	Each data beat returned to the memory controller on the DDR SDRAM interface
Cycles a read or write transfers data from (or to) DDR SDRAM	Ref:11	Each data beat transferred to or from the DDR SDRAM
Pipelined read misses in the row open table	C1:57	Row open table read misses issued while a read is outstanding
Pipelined read or write misses in the row open table	C2:0	Row open table read or write misses issued while a read or write is outstanding
Non-pipelined read misses in the row open table	C3:60	Row open table read misses issued when no reads are outstanding
Non-pipelined read or write misses in the row open table	C4:0	Row open table read or write misses issued when no reads or writes are outstanding
Pipelined read hits in the row open table	C5:56	Row open table read hits issued when a read is outstanding
Pipelined read or write hits in the row open table	C6:0	Row open table read or write hits issued when a read or write is outstanding
Non-pipelined read hits in the row open table	C7:57	Row open table read hits issued when no reads are outstanding
Non-pipelined read or write hits in the row open table	C8:0	Row open table read or write hits issued when no reads or writes are outstanding
Forced page closings not caused by a refresh	C1:0	Precharges issued to the DDR SDRAM for any reason except refresh. The possibilities are as follows: • A new transaction must be issued to an already active bank and sub-bank that has a different row open. • A new transaction must be issued, but the row open table is full and there is no bank/sub-bank match between the current transaction and the row open table. • The BSTOPRE interval expired for an open row.
Row open table misses	C2:1	Transactions that miss in the row open table
Row open table hits	C3:0	Transaction that hit in the row open table
Force page closings	C4:1	Forced page closings including those due to refreshes
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 DDR SDRAM interface.
Forced page closings due to collision with bank and sub-bank	Ref:12	Increments if a new transaction must be issued to an active bank and sub-bank that has a different row open
Reads or writes from core	Ref:13	—
Reads or writes from TSEC 1	C3:1	—
Reads or writes from TSEC 2	C4:2	—

Table 18-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Reads or writes from CPM	C5:1	—
Reads or writes from RapidIO	C3:2	—
Reads or writes from PCI	C4:3	—
Reads or writes from DMA	C5:2	—
Row open table hits for reads or writes from core	Ref:14	—
Row open table hits for reads or writes from TSEC 1	C6:1	—
Row open table hits for reads or writes from TSEC 2	C7:0	—
Row open table hits for reads or writes from CPM	C8:1	—
Row open table hits for reads or writes from RapidIO	C6:2	—
Row open table hits for reads or writes from PCI	C7:1	—
Row open table hits for reads or writes from DMA	C8:2	—
Memory Target Queue Events		
MEM TQ read/write address collision	C5:5	—
RapidIO Controller Events		
Misaligned transactions on RapidIO	C4:11	New transactions loaded into the misaligned engine
Retried packets on RapidIO transmit due to resource limitations	C2:62	Includes only packets that caused the retry, not ignored packets that must be resent
Flushed packets due to prior retries or error recovery on RapidIO transmit	C3:10	Includes the packet being acknowledge retried
Retried or not accepted packets on RapidIO Rx	C5:7	Includes only retried or unaccepted packets from a remote device, not ignored packets that must be resent
Misaligned engine priority 2 occupied	C5:12	Misaligned engine for priority 2 transactions busy
Misaligned engine priority 1 occupied	C6:12	—
Misaligned engine priority 0 occupied	C7:10	Misaligned engine for priority 0 transactions busy
ACK history queue full	C4:61	Unacked packets are outstanding.
Outbound RapidIO stopped for training event	C6:13	Event asserted during 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 inbound ACK not accepted

Table 18-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Inbound message packet protocol level retry	C4:18	—
Inbound doorbell packet protocol level retry	C6:14	—
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.
DMA Controller 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
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 double word valid
Channel 1 read DW or less	C2:7 and C6:58	DMA channel 1 read double word valid
Channel 2 read DW or less	C3:6 and C7:54	DMA channel 2 read double word valid
Channel 3 read DW or less	C4:8 and C8:52	DMA channel 3 read double word valid
Channel 0 write DW or less	C1:5	DMA channel 0 write double word valid
Channel 1 write DW or less	C2:8	DMA channel 1 write double word valid
Channel 2 write DW or less	C3:7	DMA channel 2 write double word valid
Channel 3 write DW or less	C4:9	DMA channel 3 write double word valid
e500 Coherency Module (ECM) Events		
ECM request wait core	C8:13	Asserted for every cycle core request occurs
ECM request wait CPM/SAP/boot sequencer	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 PCI/RapidIO/DMA	C4:20	Asserted for every cycle PCI request occurs

Table 18-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
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 TSEC1	C3:19	—
ECM dispatch from TSEC2	C4:21	—
ECM dispatch from RapidIO	C5:17	—
ECM dispatch from PCI	C6:17	—
ECM dispatch from DMA	C7:14	—
ECM dispatch from other	C8:14	—
ECM dispatch to DDR	C4:22	—
ECM dispatch to L2/SRAM	C5:18	—
ECM dispatch to LBC	C6:18	—
ECM dispatch to RapidIO	C7:15	—
ECM dispatch to PCI	C8:15	—
ECM dispatch snoopable	C3:20	—
ECM dispatch write	C1:17	—
ECM dispatch write allocate	C2:21	—
ECM dispatch write allocate lock	C3:21	—
ECM dispatch read	C4:23	—
ECM dispatch read unlock	C5:19	—
ECM dispatch read clear atomic	C6:19	—
ECM dispatch read set atomic	C7:16	—
ECM dispatch read decrement atomic	C8:16	—
ECM dispatch read increment atomic	C7:17	—
ECM data bus grant DDR	C1:18	—
ECM data bus grant LBC	C2:22	—
ECM data bus grant PIC	C1:19	—
ECM data bus grant CPM	C2:23	—
ECM data bus grant TSEC1	C3:23	—
ECM data bus grant TSEC2	C4:25	—
ECM data bus wait DDR	C5:20	—

Table 18-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
ECM data bus wait LBC	C6:20	—
ECM data bus wait PIC	C5:21	—
ECM data bus wait CPM	C6:21	—
ECM data bus wait TSEC1	C7:19	—
ECM data bus wait TSEC2	C8:18	—
ECM global data bus beat	Ref:16	—
ECM e500 direct read bus beat	Ref:17	—
ECM e500 direct read bus beat forwarded	C2:24	ECM direct read bus beat forwarded directly to e500 R1 data bus
ECM cancel	Ref:18	—
Interrupt Controller (PIC) Events		
PIC total interrupt count	Ref:26	Total number of interrupts serviced
PIC interrupt wait cycles	C8:62	Counts cycles when an interrupt waits to be acknowledge
PIC interrupt service cycles	C2:19	Number of cycles there is an interrupt currently being serviced
PIC interrupt select 0 (duration threshold)	C1:56	THRESHOLD: select 0–3: interrupt count over threshold. (Note: only unmasked, nonzero priority requests are acknowledged). The four interrupts are selected through register pairs, PM0MR*n*–PM3MR*n*. See Section 10.3.4, “Performance Monitor Mask Registers (PMMRs).”
PIC interrupt select 1 (duration threshold)	C3:59	
PIC interrupt select 2 (duration threshold)	C5:55	
PIC interrupt select 3 (duration threshold)	C6:60	
PCI/PCI-X Common Events		
PCI/PCI-X clock cycles	Ref:28	—
PCI/PCI-X inbound memory reads	C1:62	Includes all read types
PCI/PCI-X inbound memory writes	C2:37	—
PCI/PCI-X inbound config reads	C3:63	—
PCI/PCI-X inbound config writes	C4:37	—
PCI/PCI-X outbound memory reads	C5:30	Includes all read types
PCI outbound memory writes/PCI-X outbound memory writes attempted	C6:32	Number of PCI outbound memory writes or number of PCI-X outbound memory writes attempted
PCI/PCI-X outbound I/O reads	C3:37	—
PCI/PCI-X outbound I/O writes	C4:38	—

Table 18-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
PCI outbound config reads/PCI-X outbound config reads attempted	C7:26	Number of PCI outbound config reads OR number of PCI-X outbound config reads attempted
PCI/PCI-X outbound config writes	C8:26	—
PCI/PCI-X inbound total read data beats	C5:32	Includes 32- and 64-bit transactions
PCI/PCI-X inbound total write data beats	C6:34	Includes 32- and 64-bit transactions
PCI/PCI-X outbound total read data beats	C7:28	Includes 32- and 64-bit transactions
PCI/PCI-X outbound total write data beats	C8:28	Includes 32- and 64-bit transactions
PCI/PCI-X inbound 32-bit read data beats	C1:30	—
PCI/PCI-X inbound 32-bit write data beats	C2:38	—
PCI/PCI-X outbound 32-bit read data beats	C3:38	—
PCI/PCI-X outbound 32-bit write data beats	C4:39	—
PCI/PCI-X inbound 64-bit read data beats	C5:31	—
PCI/PCI-X inbound 64-bit write data beats	C6:33	—
PCI/PCI-X outbound 64-bit read data beats	C7:27	—
PCI/PCI-X outbound 64-bit write data beats	C8:27	—
PCI/PCI-X total transactions	C7:29	Includes 32- and 64-bit transactions
PCI/PCI-X 64-bit transactions	C8:29	—
PCI/PCI-X inbound purgeable reads	C2:2	—
PCI/PCI-X inbound (speculative reads) purgeable reads discarded	C8:63	
PCI/PCI-X idle cycles	C1:31	—
PCI/PCI-X dual address cycles	C2:40	—
PCI/PCI-X internal cycles	C3:39	—
PCI inbound memory read PCI-X inbound memory 32-bit read	C1:34	—
PCI inbound memory readline PCI-X inbound memory alias read	C2:44	—

Table 18-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
PCI inbound memory read multiple PCI-X inbound memory block read	C3:42	—
PCI outbound memory reads/PCI-X outbound memory DWORD reads attempted	C4:43	Number of PCI outbound memory reads or number of PCI-X outbound memory DWORD reads attempted
PCI outbound memory read lines/ PCI-X outbound memory burst read attempted	C5:36	Number of PCI outbound memory read lines or number of PCI-X outbound memory burst reads attempted
PCI wait PCI-X initial wait	C1:35	$\overline{\text{PCI\_IRDY}}$, $\overline{\text{PCI\_TRDY}}$ not both asserted
PCI Specific Events		
PCI cycles $\overline{\text{PCI\_IRDY}}$ is asserted	C6:36	—
PCI cycles $\overline{\text{PCI\_TRDY}}$ is asserted	C7:31	—
PCI cycles $\overline{\text{PCI\_FRAME}}$ is asserted	C8:31	—
PCI-X 1 split transaction	C2:41	Number of PCI-X cycles there are 1 or more outbound read transactions awaiting completion (either waiting for a split response, or a partially transacted dword disconnected by a SDPD, awaiting completion)
PCI-X 2 split transactions	C3:40	Number of PCI-X cycles there are 2 or more outbound read transactions awaiting completion (either waiting for a split response, or a partially transacted dword disconnected by a SDPD, awaiting completion)
PCI-X 3 split transactions	C4:41	Number of PCI-X cycles there are 3 or more outbound read transactions awaiting completion (either waiting for a split response, or a partially transacted dword disconnected by a SDPD, awaiting completion)
PCI-X 4 split transactions	C5:34	Number of PCI-X cycles there are 4 outbound read transactions awaiting completion (either waiting for a split response, or a partially transacted dword disconnected by a SDPD, awaiting completion)
PCI-X split responses	C6:37	Split responses received for an outbound transactions
PCI-X ADB disconnects	C7:32	—
PCI/PCI-X snoopable	C1:32	—
PCI/PCI-X write stash	C2:42	—
PCI/PCI-X write stash with lock	C3:41	—
PCI/PCI-X read unlock	C4:42	—
PCI/PCI-X byte enable transactions	C1:33	—
PCI/PCI-X non-byte enable transactions	C2:43	—

Table 18-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Three-Speed Ethernet Controller (TSEC) Events		
TSEC1 Address Data Filtering (ADF) Events		
Accepted frames	Ref:36	—
Individual hash table accepted frame	C7:35	—
Group hash table accepted frame	C8:35	—
Rejected frames	Ref:39	—
Dropped frames	Ref:38	Dropped frames that could have been accepted (data overflow, status overflow and lack of BDs)
Dropped frames due to data overflow	C1:41	Frames dropped because of overflow in the FIFO and 256-byte buffer
Dropped frames due to status overflow	C2:50	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	—
Transmit frames without threshold	C7:44	Transmit frames that do not hit Tx FIFO threshold
Receive FIFO above 1/4	Ref:47	—
Receive FIFO above 1/2	Ref:48	—
Receive FIFO above 3/4	Ref:49	—
DMA reads	C1:47	Descriptor and data reads
BD reads	C2:54	TxBD and RxBD reads
RxBD reads	C3:51	RxBD reads (transmit number can be calculated by subtracting this number from total)
DMA writes	C4:52	Writes (descriptor and data)
BD writes	C5:46	TxBDs and RxBDs closed
RxBD writes	C6:49	RxBDs closed (transmit number can be calculated by subtracting this number from total)
RxBD read latency (duration threshold)	Ref:41	Times RxBD read latency exceeds threshold (threshold event) Start: request asserted Stop: data acknowledge received
TxBD read latency (duration threshold)	Ref:42	Times TxBD read latency exceeds threshold (threshold event) Start: request asserted Stop: data acknowledge received
RxBD write latency (duration threshold)	Ref:43	Times RxBD write latency exceeds threshold (threshold event). Only for writes that require a response Start: request asserted Stop: end of transaction received

Table 18-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
TxBD write latency (duration threshold)	Ref:44	Times TxBD write response latency exceeds threshold (threshold event). Only for writes that require a response Start: request asserted Stop: end of transaction received
Tx data read latency (duration threshold)	Ref:45	Times data read response latency exceeds threshold (threshold event) Start: request asserted Stop: data acknowledge received
Data beats	C7:48	—
Read data beats	C8:46	—
Rx frame interrupts	C1:49	Times receive frame interrupt is set. Event occurs when the BD receive interrupt is set on a last BD
Tx frame interrupts	C3:53	Times a transmit frame interrupt is set. Event occurs when the BD transmit interrupt is set on a BD with L = 1
Rx frame processing (duration threshold)	Ref:46	Times frame processing exceeds threshold (threshold event) Start: open first BD Stop: close last BD
TSEC2 Address Data Filtering (ADF) Events		
Accepted frames	Ref:50	—
Individual hash table accepted frame	C7:36	—
Group hash table accepted frame	C8:36	—
Rejected frames	Ref:53	—
Dropped frames	Ref:52	Dropped frames that could have been accepted (data overflow, status overflow and acknowledgement of BDs)
Dropped frames due to data overflow	C1:42	Frames dropped because of overflow in the FIFO and 256-byte buffer
Dropped frames due to status overflow	C2:51	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	Receive FIFO contains receive data
Transmit frames without threshold	C7:45	Transmit frames that do not hit Tx FIFO threshold
Receive FIFO above 1/4	Ref:61	—
Receive FIFO above 1/2	Ref:62	—
Receive FIFO above 3/4	Ref:63	—
TSEC2 DMA Events		
DMA reads	C1:48	Descriptor and data reads
BD reads	C2:55	Transmit and RxBD reads

Table 18-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
RxBD reads	C3:52	Times that RxBD reads (transmit number can be calculated by subtracting this number from total)
DMA writes	C4:53	Descriptor and data writes
BD writes	C5:47	Times that TxBDs and RxBDs closed
RxBD writes	C6:50	Times RxBDs closed (transmit number can be calculated by subtracting this number from total)
RxBD read latency (duration threshold)	Ref:55	Times RxBD read latency exceeds threshold (threshold event) Start: request asserted Stop: data acknowledge received
TxBD read latency (duration threshold)	Ref:56	Times TxBD read latency exceeds threshold (threshold event) Start: request asserted Stop: data acknowledge received
RxBD write latency (duration threshold)	Ref:57	Times RxBD write response latency exceeds threshold (threshold event). Only for writes that require a response Start: request asserted Stop: End of Transaction received
TxBD write latency (duration threshold)	Ref:58	Times that TxBD write response latency exceeds threshold (threshold event). Only for writes that require a response Start: request asserted Stop: End of Transaction received
Tx data read latency (duration threshold)	Ref:59	Data read response latency exceeds threshold (threshold event) Start: request asserted Stop: Data acknowledge received
Data beats	C7:49	—
Read data beats	C8:47	—
Rx frame interrupts	C1:50	BD receive interrupt is set on a last BD
Tx frame interrupts	C3:54	BD transmit interrupt is set on a BD with L = 1
Rx frame processing	Ref:60	Receive frame processing exceeds threshold (threshold event) Start: Open first BD Stop: Close last BD
Local Bus Events		
Bank 1 hits (chip-select)	C1:51	—
Bank 2 hits (chip-select)	C2:56	—
Bank 3 hits (chip-select)	C3:55	—
Bank 4 hits (chip-select)	C4:54	—
Bank 5 hits (chip-select)	C5:48	—

Table 18-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Bank 6 hits (chip-select)	C6:53	—
Bank 7 hits (chip-select)	C7:50	—
Bank 8 hits (chip-select)	C8:50	—
Requests granted to CPM port	C1:52	—
Requests granted to ECM port	C2:57	—
Cycles atomic lock for CPM port is enabled	C3:56	—
Cycles atomic reservation for ECM port is enabled	C4:55	
Atomic reservation time-outs for CPM port	C5:49	—
Atomic reservation time-outs for ECM port	C6:54	—
Cycles a read is taking in GPCM	C1:53	—
Cycles a read is taking in UPM	C2:58	—
Cycles a read is taking in SDRAM	C3:57	—
Cycles a write is taking in GPCM	C4:56	—
Cycles a write is taking in UPM	C5:50	—
Cycles a write is taking in SDRAM	C6:55	—
SDRAM bank misses	C7:51	—
SDRAM page misses	C8:51	—
L2 Cache/SRAM Events		
Core instruction accesses to L2 that hit	Ref:22	—
Core instruction accesses to L2 that miss	C2:59	—
Core data accesses to L2 that hit	Ref:23	—
Core data accesses to L2 that miss	C4:57	—
Non-core burst write to L2 (cache external write or SRAM)	C5:51	—
Non-core non-burst write to L2	C6:56	—
Noncore write misses cache external write window and SRAM memory range	C7:52	—
Non-core read hit in L2	Ref:24	—
Non-core read miss in L2	C1:54	—

Table 18-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
L2 allocates, from any source	Ref:25	—
L2 retries due to full write queue	C2:60	—
L2 retries due to address collision	C3:58	—
L2 failed lock attempts due to full set	C4:58	—
L2 victimizations of valid lines	C5:52	—
L2 invalidations of lines	C6:57	—
L2 clearing of locks	C7:53	—
Debug Events		
External event	C3:61	Number of cycles trig_in pin is asserted
Watchpoint monitor hits	C2:61	—
Trace buffer hits	C1:58	—
Chaining Events		
PMC0 carry-out	Ref:1	PMC0[0] 1-to-0 transitions
PMC1 carry-out	Ref:2	PMC1[0] 1-to-0 transitions. Reserved for PMC1.
PMC2 carry-out	Ref:3	PMC2[0] 1-to-0 transitions. Reserved for PMC2.
PMC3 carry-out	Ref:4	PMC3[0] 1-to-0 transitions. Reserved for PMC3.
PMC4 carry-out	Ref:5	PMC4[0] 1-to-0 transitions. Reserved for PMC4.
PMC5 carry-out	Ref:6	PMC5[0] 1-to-0 transitions. Reserved for PMC5.
PMC6 carry-out	Ref:7	PMC6[0] 1-to-0 transitions. Reserved for PMC6.
PMC7 carry-out	Ref:8	PMC7[0] 1-to-0 transitions. Reserved for PMC7.
PMC8 carry-out	Ref:9	PMC8[0] 1-to-0 transitions. Reserved for PMC8.

18.4.8 Performance Monitor Examples

Table 18-12 contains sample register settings for the four supported modes.

- Simple event performance monitoring example
- Triggering event performance monitoring example
- Threshold event performance monitoring example
- Burstiness event performance monitoring example

The settings in Table 18-11 are identical for all four examples.

Table 18-11. PMGC0 and PMLCA*n* Settings

Field	Setting	Reason
PMGC0[FAC]	0	Counters must not be frozen.
PMGC0[PMIE]	1	Performance monitor interrupts are enabled.
PMGC0[FCECE]	1	Counters should be frozen when an interrupt is signalled.
PMLCA*n*[FC]	0	Counters cannot be frozen for counting.
PMLCA*n*[CE]	1	Overflow condition enable is required to allow interrupt signalling.

For simple event counting, a non-threshold event is selected in PMLCA*n*[EVENT] and all other features are disabled by clearing all register fields except for CE.

For the triggering example any event can be selected in PMLCA*n*[EVENT]. All other features are disabled by clearing these register fields except for CE to allow interrupt signalling. If PMLCB*n*[TRIGONSEL] is 3 and PMLCB*n*[TRIGOFFSEL] is 5, the counter begins and ends counting based on the conditions in counters three and five. Furthermore, if PMLCB*n*[TRIGONCNTL] is 1, the counter begins counting when PMC3 changes value. According to the setting in PMLCB*n*[TRIGOFFCNTL], the counter ends counting when PMC5 overflows. Also, although the register settings for PMC5 is not shown, PMLCA*n*[CE] for this counter must be cleared so that interrupt signalling is not enabled and the counter does not freeze when it overflows.

For threshold counting, a threshold event must be specified in PMLCA*n*[EVENT]. For this example, the duration threshold value is scaled by two because PMLCB*n*[TBMULT] is one. All other features are disabled by clearing the appropriate fields.

Any non-threshold event can use the burstiness feature. For burstiness counting, values for PMLCA*n*[BSIZE,BGRAN,BDIST] and PMLCB*n*[TBMULT] must be specified.

Table 18-12. Register Settings for Counting Examples MPC8560

Register	Register Field	Simple Event	Triggering	Threshold	Burstiness
PMGC0	FAC	0	0	0	0
	PMIE	1	1	1	1
	FCECE	1	1	1	1
PMLCA*n*	FC	0	0	0	0
	CE	1	1	1	1
	EVENT	89	68	39	2
	BSIZE	0	0	0	5
	BGRAN	0	0	0	1
	BDIST	0	0	0	8

Table 18-12. Register Settings for Counting Examples MPC8560 (continued)

Register	Register Field	Simple Event	Triggering	Threshold	Burstiness
PMLCB*n*	TRIGONSEL	0	3	0	0
	TRIGOFFSEL	0	5	0	0
	TRIGONCNTL	0	1	0	0
	TRIGOFFCNTL	0	2	0	0
	TBMULT	0	0	0	0
	THRESHOLD	0	0	3	0

Chapter 19
Debug Features and Watchpoint Facility

This chapter describes all customer-visible debug modes of the MPC8560 integrated device. The debug features on the MPC8560 pertain to these three interfaces: the local bus controller (LBC), the DDR SDRAM, and the PCI/PCI-X interface. In addition to the external interfaces, the MPC8560 provides triggering capabilities based on user-programmable events. The watchpoint and trace buffer also provide some visibility to internal buses. This chapter also describes context ID registers, useful for software debug, and describes the JTAG access port signals that comply with the IEEE 1149.1 boundary-scan specification.

19.1 Introduction

As shown in the block diagram of Figure 19-1, the MPC8560 device provides the following debug features (listed with references to sections of this chapter that describe them):

- PCI/PCI-X interface debug (Section 19.4.2, “PCI/PCI-X Interface Debug”)
- DDR SDRAM interface debug (Section 19.4.3, “DDR SDRAM Interface Debug”)
- Local bus controller (LBC) debug (Section 19.4.4, “Local Bus Interface Debug”)
- Watchpoint monitor and trace buffer debug (Section 19.4.5, “Watchpoint Monitor,” and Section 19.4.6, “Trace Buffer”)

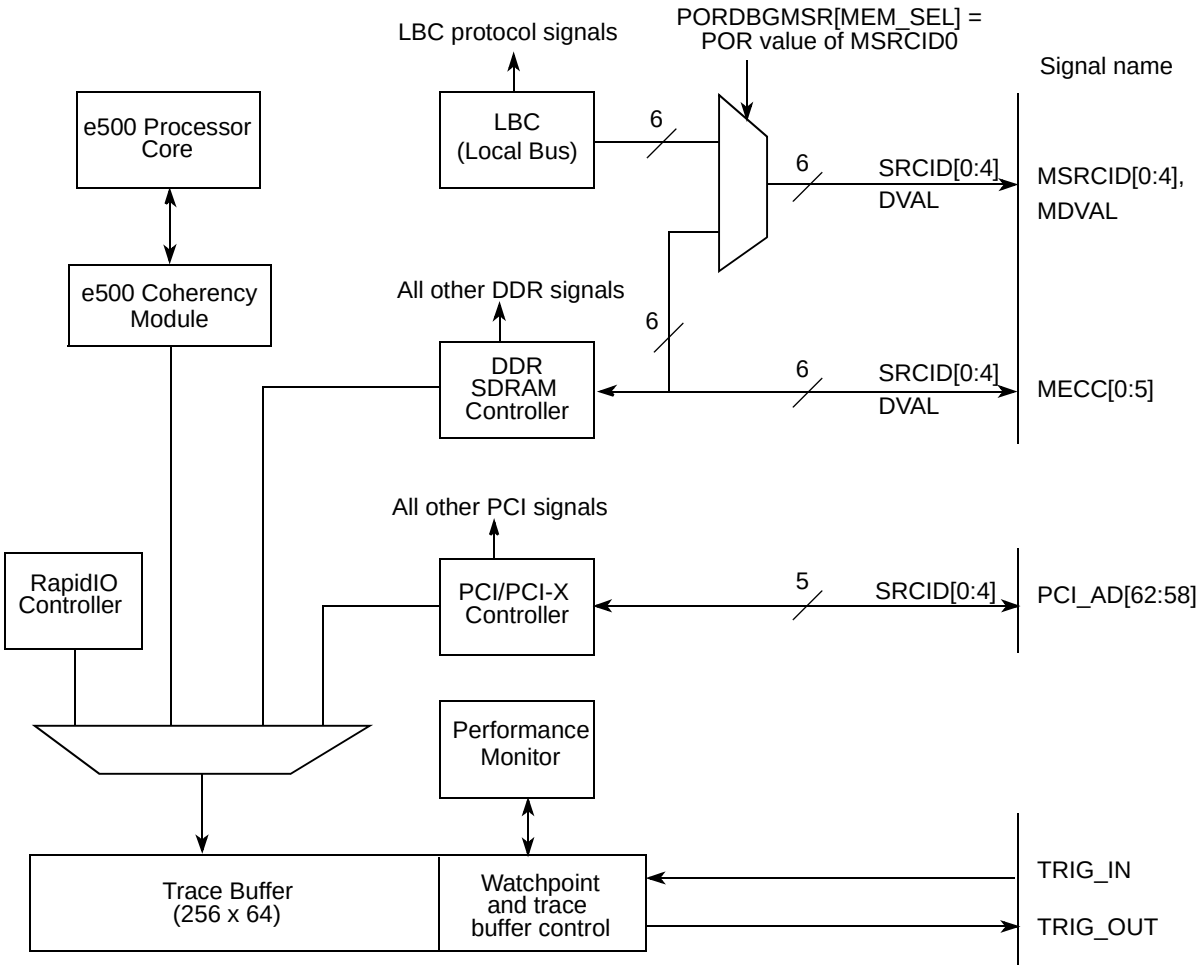


Figure 19-1. Debug and Watchpoint Monitor Block Diagram

19.1.1 Overview

As shown in Figure 19-1, debug information is provided through the following interfaces: PCI/PCI-X, LBC, and DDR SDRAM. Limited visibility, through a 256 × 64 trace buffer, is also provided for the processor core interface and on an internal RapidIO outbound interface. This visibility into internal device operation is useful for debugging application software through inverse assembly and reconstruction of the fetch stream.

The combination of a source ID (MSRCID[0:4]) and a data-valid signal (MDVAL) indicates that meaningful debug information is visible on either the local bus or DDR SDRAM interfaces. A logic analyzer can be programmed to capture data based on the values of MSRCID[0:4] and MDVAL.

Other system debugging is supported by the programmable triggering of the watchpoint monitor and trace buffer. Both can be triggered from one of the following three sources:

- Each other
- A performance monitor event
- An external source (through TRIG_IN)

The watchpoint monitor can be configured to assert TRIG_OUT when a programmed event occurs. The two context ID registers, described in Section 19.3.3, “Context ID Registers,” are useful for software debug.

19.1.2 Features

The principle features of the debug modes and the watchpoint monitor are as follows:

- PCI/PCI-X interface: transaction source ID driven onto PCI_AD[62:58]
- LBC and DDR interface source ID and data-valid indicators
  - LBC or DDR SDRAM source ID can be selected to be driven onto MSRCID[0:4]
  - Source ID and data-valid indicators can be selected to be driven onto the error correcting code (ECC) pins of the DDR interface
- Watchpoint monitor that supports the following:
  - Two-level triggering
  - Programmable external trigger (TRIG_OUT)
  - Interlocking with performance monitor to use its large number of counters
- Trace buffer features that supports the following:
  - Two-level triggering
  - Programmable external trigger (TRIG_OUT)
  - Interlocking with performance monitor to use its large number of counters
  - 256-entry trace buffer, 64 bits each
  - Programmable trace start and stop
  - Functionality as a second watchpoint monitor
- Context ID registers that can be programmed to trigger events

19.1.3 Modes of Operation

The PCI/PCI-X, LBC, and DDR SDRAM interfaces all have debug modes, which are controlled by values on configuration inputs during the power-on reset (POR) sequence, as shown in Table 19-3. The DDR controller can also drive debug information on either MSRCID[0:4] or

MECC[0:5]. See Section 19.4.1, “Source and Target ID,” for additional information about the source ID information driven on the debug signals in these modes.

Note that both the watchpoint monitor and trace buffer also operate in a variety of modes.

Table 19-1. POR Configuration Settings and Debug Modes

Configuration Signal	POR Value	Effect	Reference
MSRCID0	0	Local bus SDRAM information appears on MSRCID[0:4] and MDVAL.	19.1.3.1/19-4
	1	Default value (internal pull-up resistor). DDR SDRAM information appears on MSRCID[0:4] and MDVAL.	
MSRCID1	0	MECC[0:4] operate in debug mode and provide memory debug source ID and MECC5 provides data-valid information.	19.1.3.2/19-4
	1	Default value (internal pull-up resistor). MECC[0:4] operate in normal mode and provide DDR SDRAM error correcting code information.	
$\overline{\text{PCI\_GNT3}}$	0	The PCI/PCI-X interface operates in debug mode. The source ID information appears on the high-order address bits (PCI_AD[62:58]) during the bus command phase.	19.1.3.3/19-5
	1	Default value (internal pull-up resistor). The PCI/PCI-X interface operates in normal mode.	

19.1.3.1 Local Bus (LBC) Debug Mode

The LBC and the DDR SDRAM controller can drive debug information (source ID and data-valid indicator) onto MSRCID[0:4] and MDVAL As shown in Table 19-1, the MSRCID0 value during POR controls multiplexing. If MSRCID0 is low when sampled during POR, the local bus SDRAM information appears on MSRCID[0:4] and MDVAL; otherwise, the DDR SDRAM debug information is presented.

19.1.3.2 DDR SDRAM Interface Debug Modes

MSRCID1 is sampled during POR to multiplex either ECC or debug information on the ECC pins of the DDR SDRAM interface. As shown in Table 19-1, if MSRCID1 is low during POR, the ECC pins operate in debug mode and provide memory debug source ID and data-valid information. MSRCID1 must be pulled low during POR to use the ECC pins in debug mode. If MSRCID1 is unconnected, an internal pull-up resistor ensures the ECC pins always source DDR SDRAM error correcting code information as their default power-on reset configuration.

NOTE

If the DDR ECC pins are in debug mode (configured for debug during POR), ECC checking is disabled in the memory controller. In this case, MECC[0:4] do not provide ECC information and must not be connected to SDRAM devices.

19.1.3.3 PCI/PCI-X Interface Debug Modes

If $\overline{\text{PCI\_GNT3}}$ is low when sampled during POR, the PCI/PCI-X interface operates in debug mode. In this mode, the source ID information appears on the high-order address bits (PCI_AD[62:58]) during the bus command phase. See Section 19.4.2, “PCI/PCI-X Interface Debug,” for more information.

19.1.3.4 Watchpoint Monitor Modes

The watchpoint monitor supports the following operating modes:

- Immediate trigger arming (one-level triggering)—The watchpoint monitor triggers as soon as the first trigger event occurs.
- Wait for trigger arming (two-level triggering)—The watchpoint monitor waits for a specific event before enabling (arming) the trigger logic. The monitor does not respond to trigger events until after the arming event occurs. This function is similar to two-level triggering on a logic analyzer.
- Assert TRIG_OUT on hit—The debug block can be programmed to assert the TRIG_OUT signal when a programmed watchpoint monitor event occurs. This signal can be used to trigger a logic analyzer.

19.1.3.5 Trace Buffer Modes

The trace buffer supports the following operating modes:

- Immediate trigger arming (one-level triggering)—The trace buffer triggers as soon as the first trigger event occurs.
- Wait for trigger arming (two-level triggering)—The trace buffer waits for a specific event before enabling (arming) the trigger logic. The trace buffer does not respond to trigger events until after the arming event occurs. This function is similar to two-level triggering on a logic analyzer.
- Specific interface selection—The trace buffer can be programmed to trace one of several internal interfaces.
- Specific event selection—The trace buffer can be programmed to trace on the occurrence of one or several concurrent events.
- Specific trace selection—To facilitate trace data filtering, the trace buffer can be configured to capture data under the following conditions:
  - On every cycle in which a valid transaction is present on the selected interface
  - Only when a watchpoint monitor event occurs
  - Only when the programmed trace event is detected
- Programmable trace stop—The trace buffer may be programmed to stop tracing when a programmed stop-tracing event occurs or when the 256-entry buffer is full.

19.2 External Signal Descriptions

This section provides information about all the external signals associated with the various MPC8560 debug functions.

As shown in Table 19-1, the MPC8560 has several signals that are sampled during POR to determine the configuration of the phase-locked loop clock mode and the ROM, Flash, and dynamic memory. See Chapter 4, "Reset, Clocking, and Initialization."

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 JTAG TAP signals.

19.2.1 Overview

All the signals associated with device debug features are summarized in Table 19-2, listed with a reference to the page number of the section with more information. The detailed descriptions are contained in Table 19-2. Some signals (the MECC bus for example) are additionally described in other chapters, but are described here also for completeness, with emphasis on their debugging utility.

Table 19-2. Debug, Watchpoint and Test Signal Summary

Name	Description	Functional Block	Function	Reset Value	I/O	Page #
MDVAL	Memory data-valid	Debug	Selectable data-valid signal from either DDR SDRAM controller or LBC	1	O	19-7
MECC[0:7]	DDR error correcting code	DDR SDRAM	In debug mode, the 6 high-order bits carry debug information (transaction source ID and data-valid indication).	0x08	O[1]	19-8
MSRCID[0:1]	Memory source ID	Debug	Selectable transaction source ID from either DDR SDRAM controller or local bus controller	Reset_cfg	O	19-8
MSRCID[2:4]				111	O	19-8
TRIG_IN	Trigger in	Debug	Trigger for various function in the watchpoint monitor and trace buffer	1	I	19-8
TRIG_OUT	Trigger out	Debug	Can be used externally for triggering a logic analyzer. Additionally, it can be used for observing system ready indication. Functions are multiplexed onto this signal depending on TOSR[SEL] (see Table 19-25).	1	O	19-8
PCI_AD[62:58]	PCI address	Debug	In debug mode these pins carry the source ID of the transaction.	0000_0	O	19-8
TCK	Test clock	Debug	Clock for JTAG testing. Internally pulled up	1	I	19-9
TDI	Test data input	Debug	Serial input for instructions and data to the JTAG test subsystem. Internally pulled up	1	I	19-9
TDO	Test data output	Debug	Serial data output for the JTAG test subsystem. High impedance except when scanning out data	Hi-Z	O	19-9

Table 19-2. Debug, Watchpoint and Test Signal Summary (continued)

Name	Description	Functional Block	Function	Reset Value	I/O	Page #
TMS	Test mode select	Debug	Carries commands to the TAP controller for boundary scan operations. Internally pulled up	1	I	19-9
$\overline{\text{TRST}}$	Test reset	Debug	Resets the TAP controller asynchronously	—	I	19-9
THERM[0:1]	Thermal resistor access	Test	These pins tie directly to an internal resistor whose value varies linearly with temperature.	—	I	19-9
$\overline{\text{TEST\_SEL}}$	Test select	Test	Factory test. Must be negated for normal operation	—	I	19-9
$\overline{\text{LSSD\_MODE}}$	Test	Test	Factory Test. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper treatment.		I	19-9
L1_TSTCLK	Test	Test	Factory Test. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper treatment.		I	19-9
L2_TSTCLK	Test	Test	Factory Test. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper treatment.		I	19-9

1 Although these signals are normally bidirectional, when sourcing debug information they are output only.

19.2.2 Detailed Signal Descriptions

This section describes the details of the debug, watchpoint monitor, and JTAG test signals

19.2.2.1 Debug Signals—Details

Table 19-3 describes all signals associated with device debug modes.

Table 19-3. Debug Signals—Detailed Signal Descriptions

Signal	I/O	Description	
MDVAL	O	Memory data-valid. Indicates when valid data is available. May be used by a logic analyzer to capture the data on the data bus.	
		State Meaning	Asserted—Indicates that data is valid on the data bus during the current clock cycle. When the DDR SDRAM interface is selected to source information on MDVAL, this signal is valid for every cycle that data is driven or received on the DDR SDRAM interface. When the LBC is selected, this signal is valid for every cycle that data is driven or received on the local bus interface. The assertion of this signal may be used by a logic analyzer to capture data.
		Timing	Asserted/Negated—Referenced to the selected interface, (DDR or local bus). Asserts when data is valid. Assertions are held for the duration of the transfer. Read data timing is similar to MA. Write data timing is similar to the output MDQ.

Table 19-3. Debug Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
MECC[0:7]	O	Memory ECC. DDR error checking and correcting. The normally bidirectional operation of the memory ECC (MECC) bus is described in Section 9.5.11, "Error Checking and Correcting (ECC)." This bus is used for debug functions when MSRCID1 is sampled low during POR. In debug mode, the high-order 5 bits (MECC[0:4]) may be used to provide the transaction source ID and MECC5 can be used as the data-valid indicator. In debug mode, MECC[0:5] is constantly driven with debug information and must be disconnected from the DDR memory's ECC pins.	
		State Meaning	Asserted/Negated—In debug mode, MECC[0:5] is always driven. The source ID values appear during RAS and CAS cycles. A value of 0x1F (all ones) is driven during cycles other than RAS and CAS. The data-valid indicator appears when data is being received or driven on the pins.
		Timing	Driven every cycle in debug mode
MSRCID[0:4]	O	Memory source ID. Attribute signals associated with the memory interface that indicate the source ID for a transaction on an SDRAM interface. The SDRAM interface, DDR or local bus, to which the debug information applies is specified during POR with MSRCID0 as shown in Table 19-1. Two of these signals serve as reset configuration input signals.	
		State Meaning	Asserted/Negated—In debug mode, always driven with the value of the source ID. The source ID has a value of 0x1F for cycles other than RAS and CAS. The encodings shown in Table 19-26 provide detailed information about a memory transaction.
		Timing	Driven every cycle in debug mode. Similar timing to MA
PCI_AD[62:58]	O	PCI Address. Provides transaction source ID for the current PCI bus transaction	
		State Meaning	Asserted/Negation—In debug mode, always driven with the value of the transaction source ID
		Timing	Driven every cycle in debug mode

19.2.2.2 Watchpoint Monitor Trigger Signals—Details

Table 19-4 shows detailed descriptions of the watchpoint monitor and trace buffer signals.

Table 19-4. Watchpoint and Trigger Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TRIG_IN	I	Trigger in. Can be used to trigger the watchpoint and trace buffers. Note this is an active-high (rising-edge triggered) signal.	
		State Meaning	Asserted—Indicates that a programmed/armed external event has been detected. Assertion may be used internally to trigger trace buffers and watchpoint mechanisms.
		Timing	Assertion/Negation—The MPC8560 interprets TRIG_IN as asserted on detection of the rising edge. It may occur at any time. Must remain asserted for at least 3 system clocks to be recognized internally

Table 19-4. Watchpoint and Trigger Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
TRIG_OUT	O	Trigger out. Function determined by TOSR[SEL]. When TOSR[SEL] is non-zero, it can be used for triggering external devices, like a logic analyzer, with either the watchpoint monitor, the trace buffer, or the performance monitor as trigger sources. When TOSR[SEL] is cleared, TRIG_OUT is multiplexed with READY, which indicates the operational readiness of the device (running or in low-power or debug modes). See Chapter 4, “Reset, Clocking, and Initialization,” and Chapter 17, “Global Utilities,” for more details about reset, low power, and debug states.	
		State Meaning	Asserted—When TOSR[SEL] is all zeros, serves as the READY signal, indicating that the device is not in a low-power or debug mode and that it has emerged from reset. SEL ≠ 0 indicates that a programmed trigger event has occurred. Negation—No final watchpoint match condition
		Timing	Assertion may occur at any time. Remains asserted for at least 3 system clocks

19.2.2.3 Test Signals—Details

Table 19-5 shows detailed descriptions of the JTAG test signals.

Table 19-5. JTAG Test and Other Signals—Detailed Signal Descriptions

Signal	I/O	Description	
TCK	I	JTAG test clock	
		State Meaning	Asserted/Negated—Should be driven by a free-running clock signal with a 30–70% duty cycle. Input signals to the TAP are clocked in on the rising edge. Changes to the TAP output signals occur on the falling edge. The test logic allows TCK to be stopped. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 standard for more details
TDI	I	JTAG test data input	
		State Meaning	Asserted/Negated—The value present on the rising edge of TCK is clocked into the selected JTAG test instruction or data register. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 standard for more details
TDO	O	JTAG test data output	
		State Meaning	Asserted/Negated—The contents of the selected internal instruction or data register are shifted out on this signal on the falling edge of TCK. Remains in a high-impedance state except when scanning data.
		Timing	See IEEE 1149.1 standard for more details
TMS	I	JTAG test mode select	
		State Meaning	Asserted/Negated—Decoded by the internal JTAG TAP controller to distinguish the primary operation of the test support circuitry. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor.
		Timing	See IEEE 1149.1 standard for more details

Table 19-5. JTAG Test and Other Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
TRST	I	JTAG test reset	
		State Meaning	Asserted—Causes asynchronous initialization of the internal JTAG TAP controller. Must be asserted during power-on reset in order to properly initialize the JTAG TAP and for normal operation of the MPC8560. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor. Negated— Normal operation
		Timing	See IEEE 1149.1 standard for more details
LSSD_MODE	I	Used for factory test. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper treatment.	
L1_TSTCLK	I	Used for factory test. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper treatment.	
L2_TSTCLK	I	Used for factory test. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper treatment.	
THERM[0:1]	I	These signals provide access to an internal resistor that has a value that varies linearly with temperature. The actual value for the resistor varies from device to device, but the linear relationship between temperature and resistance is consistent. See the *MPC8560 Integrated Processor Hardware Specifications* for more information on how to accurately measure the junction temperature of a device. Note that this thermal resistor is intended for engineering development only.	
TEST_SEL	I	Used for factory test. Must be negated for normal operation	

19.3 Memory Map/Register Definition

Table 19-6 shows the memory-mapped debug and watchpoint registers of the MPC8560.

Table 19-6. Debug and Watchpoint Monitor Memory Map

Local Memory Offset	Register	Access	Reset	Section/Page
Watchpoint Monitor Registers				
0xE_2000	WMCR0—Watchpoint monitor control register 0	R/W	0x0000_0000	19.3.1.1/19-11
0xE_2004	WMCR1—Watchpoint monitor control register 1	R/W	0x0000_0000	19.3.1.1/19-11
0xE_200C	WMAR—Watchpoint monitor address register	R/W	0x0000_0000	19.3.1.2/19-13
0xE_2014	WMAMR—Watchpoint monitor address mask register	R/W	0x0000_0000	19.3.1.3/19-14
0xE_2018	WMTMR—Watchpoint monitor transaction mask register	R/W	0x0000_0000	19.3.1.4/19-14
0xE_201C	WMSR—Watchpoint monitor status register	R/W	0x0000_0000	19.3.1.5/19-16
Trace Buffer Registers				
0xE_2040	TBCR0—Trace buffer control register 0	R/W	0x0000_0000	19.3.2.1/19-16
0xE_2044	TBCR1—Trace buffer control register 1	R/W	0x0000_0000	19.3.2.1/19-16

Table 19-6. Debug and Watchpoint Monitor Memory Map (continued)

Local Memory Offset	Register	Access	Reset	Section/Page
0xE_204C	TBAR—Trace buffer address register	R/W	0x0000_0000	19.3.2.2/19-19
0xE_2054	TBAMR—Trace buffer address mask register	R/W	0x0000_0000	19.3.2.3/19-19
0xE_2058	TBTMR—Trace buffer transaction mask register	R/W	0x0000_0000	19.3.2.4/19-20
0xE_205C	TBSR—Trace buffer status register	R/W	0x0000_0000	19.3.2.5/19-21
0xE_2060	TBACR—Trace buffer access control register	R/W	0x0000_0000	19.3.2.6/19-22
0xE_2064	TBADHR—Trace buffer access data high register	R/W	0x0000_0000	19.3.2.7/19-22
0xE_2068	TBADR—Trace buffer access data register	R/W	0x0000_0000	19.3.2.8/19-23
Context ID Registers				
0xE_20A0	PCIDR—Programmed context ID register	R/W	0x0000_0000	19.3.3.1/19-24
0xE_20A4	CCIDR—Current context ID register	R/W	0x0000_0000	19.3.3.2/19-24
Other Registers				
0xE_20B0	TOSR—Trigger output source register	R/W	0x0000_0000	19.3.4.1/19-25

19.3.1 Watchpoint Monitor Register Descriptions

The following sections describe the control registers for the watchpoint monitor facility.

19.3.1.1 Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1)

The watchpoint monitor control registers (WMCR0, WMCR1) shown in Figure 19-2 and Figure 19-3 control the specification of watchpoint monitor events.

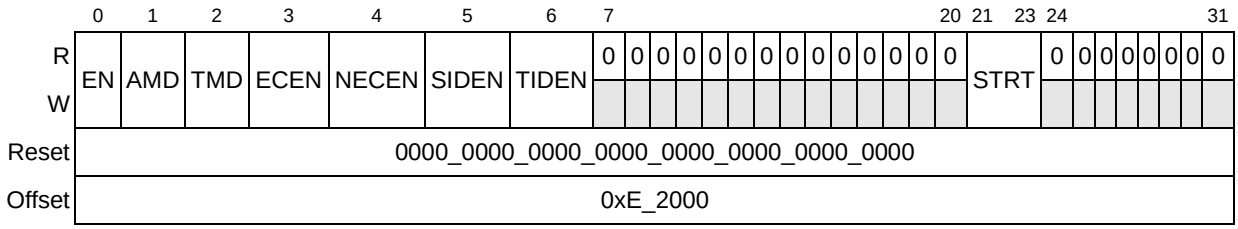


Figure 19-2. Watchpoint Monitor Control Register 0 (WMCR0)

Table 19-7 describes WMCR0 fields.

Table 19-7. WMCR0 Field Descriptions

Bits	Name	Description
0	EN	Enable 0 Watchpoint monitor events are not flagged. 1 A watchpoint monitor event is flagged.
1	AMD	Address match disable. Qualifies address match as a watchpoint event criterion. 0 Address matching is used to recognize a watchpoint event. 1 Address matching does not affect watchpoint event detection.
2	TMD	Transaction match disable. Qualifies transaction type match (as defined in WMCR1[IFSEL] and WMTMR) as a watchpoint event criterion. 0 A transaction type match is used to recognize watchpoint events. 1 A transaction type match does not affect watchpoint event detection.
3	ECEN	Equal context enable. Qualifies the matching of current context with programmed context as a watchpoint event criterion, as written in the context registers described in Section 19.3.3, “Context ID Registers.” 0 Current context match does not affect watchpoint event detection 1 Watchpoint events are qualified by comparing current context with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
4	NECEN	Not equal context enable. Qualifies the matching of current context with programmed context as a watchpoint event criterion, as written in the context registers described in Section 19.3.3, “Context ID Registers.” 0 The failure of a current context match does not affect watchpoint event detection 1 Watchpoint events are qualified with NOT getting a current context compare with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
5	SIDEN	Source ID enable 0 Source ID does not affect watchpoint event detection 1 Watchpoint events are qualified by comparison with the programmed WMCR1(SID) value.
6	TIDEN	Target ID enable 0 Target ID does not affect watchpoint event detection 1 Watchpoint events are qualified by comparison with the programmed WMCR1(TID) value.
7–20	—	Reserved
21–23	STRT	Start condition. Specifies the event that arms the watchpoint monitor to start looking for the programmed event. 000 No event. Armed immediately 001 Trace buffer event is detected 010 Performance monitor signals overflow 011 TRIG_IN transitions from 0 to 1. 100 TRIG_IN transitions from 1 to 0. 101 Current context ID equals programmed context ID 110 Current context ID is not equal to programmed context ID 111 Reserved
24–31	—	Reserved

Figure 19-3 shows the WMCR1.

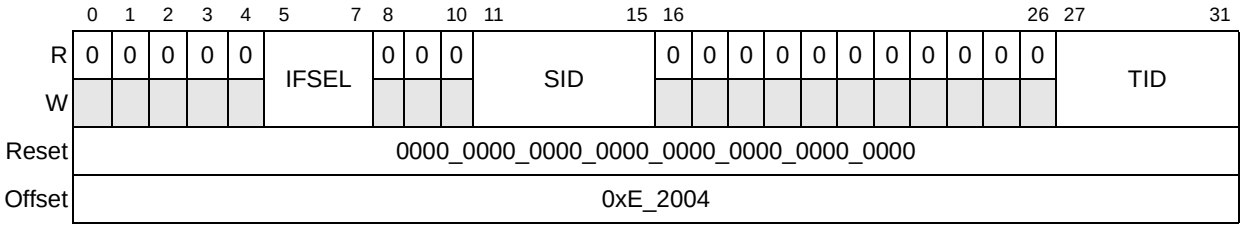


Figure 19-3. Watchpoint Monitor Control Register 1 (WMCR1)

Table 19-8 describes the WMCR1 fields.

Table 19-8. WMCR1 Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	IFSEL	Interface selection. Selects the address, transaction type (as defined in WMTMR), and other attributes to be used for comparison 000 Select e500 coherency module (ECM) dispatch interface 001 Select internal DDR SDRAM interface 010 Select internal PCI outbound interface 011 Select internal RapidIO interface 1xx Reserved
8–10	—	Reserved
11–15	SID	Source ID. Specifies the source ID associated with WMCR0[SIDEN]. For a definition of the source ID, see Table 19-26.
16–26	—	Reserved
27–31	TID	Target ID. Specifies the target ID associated with WMCR0[TIDEN]. For a definition of the target ID, see Table 19-26.

19.3.1.2 Watchpoint Monitor Address Register (WMAR)

The watchpoint monitor address register (WMAR) shown in Figure 19-4 contains the address to match against if WMCR[AMD] is clear. Note that the transaction address is qualified with the bits described in Section 19.3.1.3, “Watchpoint Monitor Address Mask Register (WMAMR),” before being compared with WMAR. Note also that the contents of WMAR are not qualified with WMAMR.

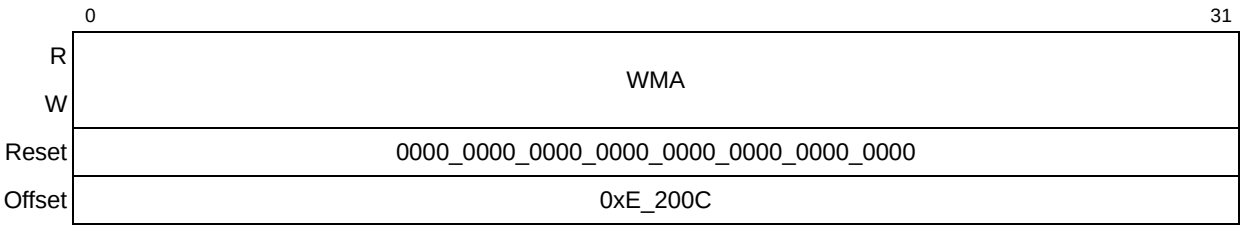


Figure 19-4. Watchpoint Monitor Address Register (WMAR)

Table 19-9 describes the WMAR fields.

Table 19-9. WMAR Field Descriptions

Bits	Name	Description
0–31	WMA	Watchpoint monitor address.

19.3.1.3 Watchpoint Monitor Address Mask Register (WMAMR)

The watchpoint monitor address mask register (WMAMR) shown in Figure 19-5 contains the mask that is applied to a transaction address before the address is compared with WMAR.

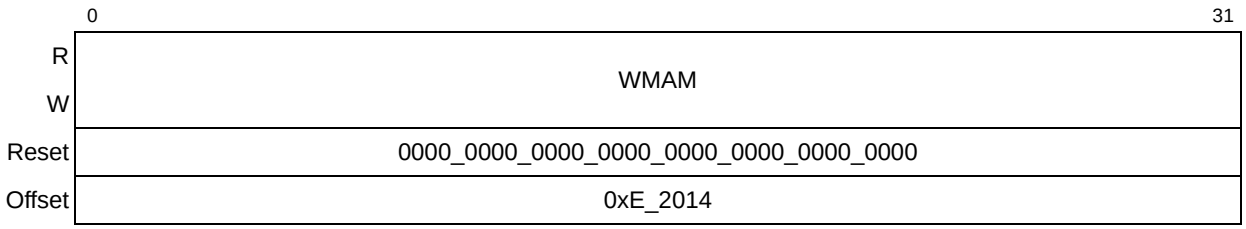


Figure 19-5. Watchpoint Monitor Address Mask Register (WMAMR)

Table 19-10 describes the WMAMR fields.

Table 19-10. WMAMR Field Descriptions

Bits	Name	Description
0–31	WMAM	Watchpoint monitor address mask. A value of zero masks the address comparison for the corresponding address bit.

19.3.1.4 Watchpoint Monitor Transaction Mask Register (WMTMR)

The watchpoint monitor transaction mask register (WMTMR), shown in Figure 19-6, specifies which transaction types to monitor. WMTMR allows users to qualify watchpoint events specifically with any combination of transaction types. As shown in Table 19-12, each bit represents as many as four separate transaction types; one for each interface. Setting a bit enables watchpoint monitoring for the corresponding transaction types.

Because the supported transaction types vary by interface, the type designated by a WMTMR field also depends on the interface specified by WMCR1[IFSEL]. Table 19-12 lists transaction types associated with each WMTMR bit by interface.

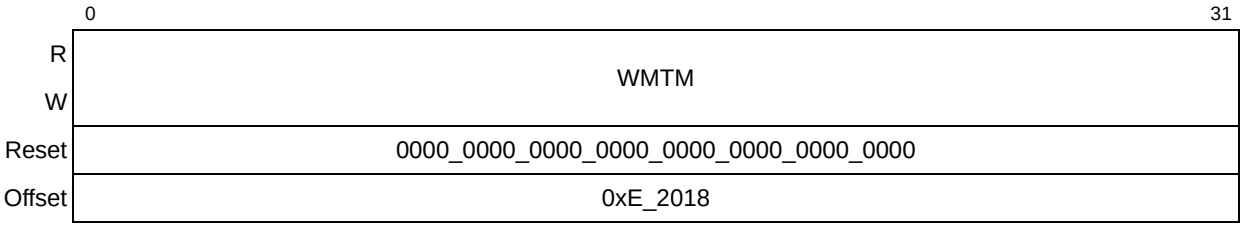


Figure 19-6. Watchpoint Monitor Transaction Mask Register (WMTMR)

Table 19-11 describes the WMTMR fields.

Table 19-11. WMTMR Field Descriptions

Bits	Name	Description
0–31	WMTM	Watchpoint monitor transaction mask. Each bit corresponds to a transaction type as defined in Table 19-12. The transaction associated with any particular bit may be different depending on the interface being monitored. A value of 1 for a given mask bit enables the matching of the transaction associated with that bit. These bits are meaningful only when WMCR0[TMD]=0.

The following table defines the transactions associated with each transaction mask bit for the different interfaces supported by the watchpoint monitor.

Table 19-12. Transaction Types by Interface

Bit	e500 Coherency Module Dispatch	DDR Controller	PCI Outbound Request	RapidIO Outbound Request
0	Write with local processor snoop	Write	Memory write	NWRITE
1	Write with no local processor snoop	—	I/O write	NWRITE_R
2	Write with allocate (L2 stashing)	Write with allocate	—	SWRITE
3	Write with allocate and lock (L2 stashing with locking)	Write with allocate and lock	—	FLUSH with data
4–7	Reserved			
8	Read with local processor snoop	Read	Memory Read	NREAD
9	Read with no local processor snoop	—	I/O Read	IO_READ_HOME
10	Read with unlock	Read with unlock	—	—
11–15	Reserved			
16	Atomic clear	ATOMIC clear	—	ATOMIC clear
17	Atomic set	ATOMIC set	—	ATOMIC set
18	Atomic decrement	ATOMIC decrement	—	ATOMIC decrement
19	Atomic increment	ATOMIC increment	—	ATOMIC increment
20–27	Reserved			
28	—	—	—	MAINTENANCE write

Table 19-12. Transaction Types by Interface (continued)

Bit	e500 Coherency Module Dispatch	DDR Controller	PCI Outbound Request	RapidIO Outbound Request
29	—	—	—	MAINTENANCE read
30	—	—	—	DOORBELL
31	—	—	—	MESSAGE

19.3.1.5 Watchpoint Monitor Status Register (WMSR)

The watchpoint monitor status register (WMSR) shown in Figure 19-7 indicates the state of the watchpoint monitor.

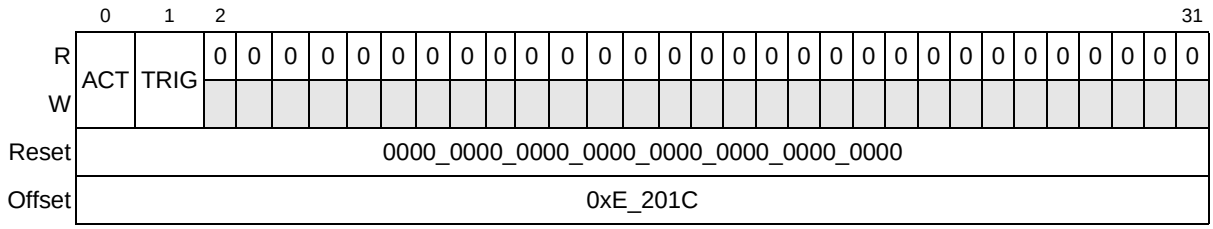


Figure 19-7. Watchpoint Monitor Status Register (WMSR)

Table 19-13 describes the WMSR fields.

Table 19-13. WMSR Field Descriptions

Bits	Name	Description
0	ACT	Active 0 The start triggering event has not occurred; watchpoint monitor is not armed. 1 The start triggering event has occurred; watchpoint monitor is armed.
1	TRIG	Triggered 0 The programmed event in WMCR0 has not yet been triggered. 1 The programmed event in WMCR0 has been triggered at least once.
2–31	—	Reserved

19.3.2 Trace Buffer Register Descriptions

The following sections describes the trace buffer registers.

19.3.2.1 Trace Buffer Control Registers (TBCR0, TBCR1)

The trace buffer control registers (TBCR0, TBCR1), shown in Figure 19-8 and Figure 19-9, specify trace buffer events.

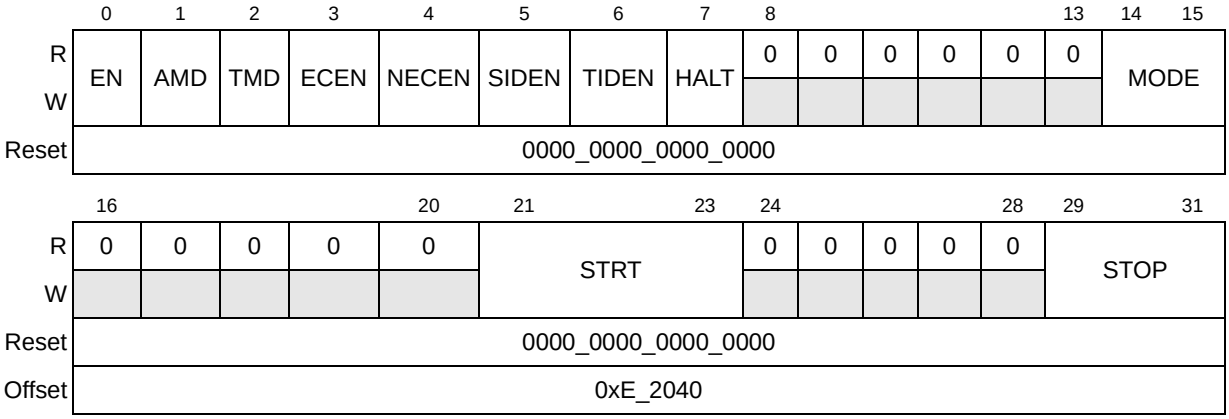


Figure 19-8. Trace Buffer Control Register 0 (TBCR0)

Table 19-14 describes the TBCR0 fields.

Table 19-14. TBCR0 Field Descriptions

Bits	Name	Description
0	EN	Enable 0 The trace buffer facility is disabled. 1 The trace buffer facility is enabled.
1	AMD	Address match disable 0 The address match is used to qualify a trace buffer event. 1 The address match is ignored when detecting a trace buffer event.
2	TMD	Transaction match disable 0 The transaction type match is used to qualify a trace buffer event. 1 The transaction type match is ignored when detecting a trace buffer event.
3	ECEN	Equal context enable. Qualifies the matching of current context with programmed context as a trace buffer event criterion, as written in the context registers described in Section 19.3.3, “Context ID Registers.” 0 Current context match does not affect trace buffer event detection. 1 Trace buffer events are qualified by comparing current context with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
4	NECEN	Not equal context enable. Qualifies the matching of current context with programmed context as a trace buffer event criterion, as written in the context registers described in Section 19.3.3, “Context ID Registers.” 0 The failure of a current context match does not affect trace buffer event detection 1 trace buffer events are qualified with NOT getting a current context compare with the programmed context event value. **Note:** ECEN and NECEN must not be enabled in the same run. If both are set, watchpoint events are inhibited (never occur).
5	SIDEN	Source ID enable 0 Trace buffer events ignore the programmed source ID value. 1 Trace buffer events are qualified by comparison with the programmed SID event value.
6	TIDEN	Target ID enable 0 Trace buffer events ignore the programmed TID event value. 1 Trace buffer events are qualified by comparison with the programmed TID event value. This comparison only applies when the ECM is selected for tracing (TBCR1[IFSEL] is all zeros).

Table 19-14. TBCR0 Field Descriptions (continued)

Bits	Name	Description
7	HALT	Halt causes the trace buffer to stop tracing immediately.
8–13	—	Reserved
14–15	MODE	Trace mode. Specifies one of two trace modes 00 Trace every valid transaction 01 Reserved 10 Trace only cycles in which a trace event is detected. Note that if EN and other TBCR0 fields are not properly programmed to specify a traceable event, tracing occurs for every valid address. 11 Reserved
16–20	—	Reserved
21–23	STRT	Start condition. Specifies the event that arms the trace buffer to start looking for the programmed event. 000 No event. Armed immediately 001 Watchpoint monitor event is detected 010 Trace buffer event is detected 011 Performance monitor signals overflow 100 TRIG_IN transitions from 0 to 1. 101 TRIG_IN transitions from 1 to 0. 110 Current context ID equals programmed context ID 111 Current context ID does not equal programed context ID
24–28	—	Reserved
29–31	STOP	Trace stop mode. Specifies the event that stops the updating of the trace buffer after it has been started. Trace buffer only stops after it has been triggered at least once. 000 Buffer is full 001 Watchpoint monitor event is detected 010 Trace buffer event is detected 011 Performance monitor signals overflow 100 TRIG_IN transitions from 0 to 1. 101 TRIG_IN transitions from 1 to 0. 110 Current context ID equals programmed context ID 111 Current context ID does not equal programed context ID

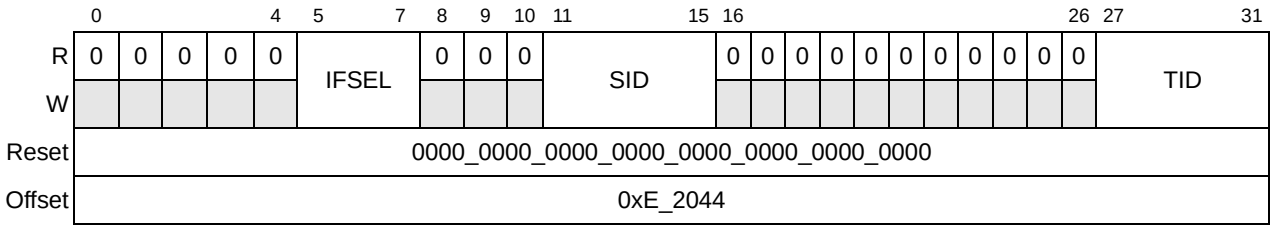


Figure 19-9. Trace Buffer Control Register 1 (TBCR1)

Table 19-15 describes the TBCR1 fields.

Table 19-15. TBCR1 Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	IFSEL	Interface selection. Specifies the interface that sources information for both comparison/buffer control and buffer data capture 000 Selects e500 coherency module (ECM) dispatch interface 001 Selects internal DDR SDRAM interface 010 Selects internal PCI outbound interface 011 Selects internal RapidIO interface 1xx Reserved
8–10	—	Reserved
11–15	SID	Source ID. Specifies the source ID associated with TBCR0[SIDEN]. The source ID is defined in Table 19-26.
16–26	—	Reserved
27–31	TID	Target ID. Specifies the target ID associated with TBCR0[TIDEN]. The target ID is defined in Table 19-26.

19.3.2.2 Trace Buffer Address Register (TBAR)

The trace buffer address register (TBAR) shown in Figure 19-10 contains the address to match against (if TBCR0[AMD] is zero). The transaction address is qualified with the bits described in Section 19.3.2.3, “Trace Buffer Address Mask Register (TBAMR),” before being compared with TBAR. Note that the contents of TBAR are not qualified with TBAMR.

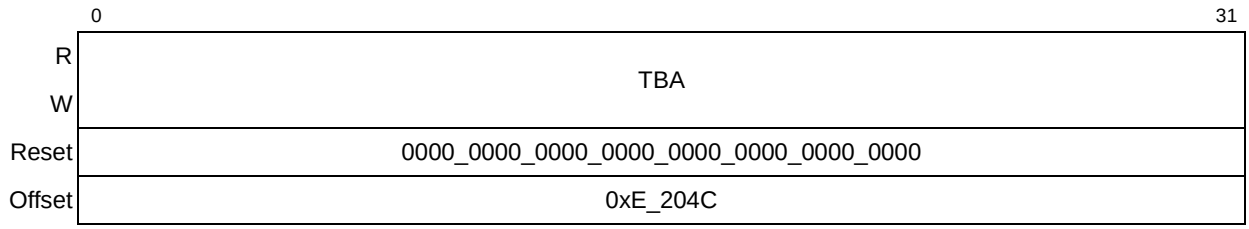


Figure 19-10. Trace Buffer Address Register (TBAR)

Table 19-16 describes the TBAR field.

Table 19-16. TBAR Field Descriptions

Bits	Name	Description
0–31	TBA	Trace buffer address

19.3.2.3 Trace Buffer Address Mask Register (TBAMR)

The trace buffer address mask register (TBAMR) shown in Figure 19-11 contains the mask that is applied to a transaction address before the address is compared with TBAR.

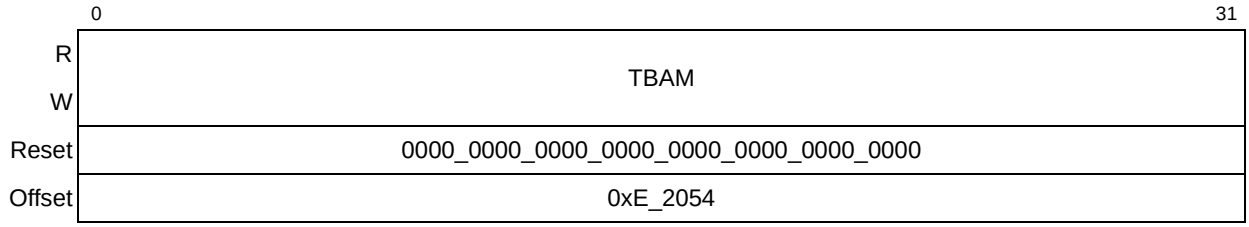


Figure 19-11. Trace Buffer Address Mask Register (TBAMR)

Table 19-17 describes the TBAMR field.

Table 19-17. TBAMR Field Descriptions

Bits	Name	Description
0–31	TBAM	Trace buffer address mask.A value of zero masks the address comparison for the corresponding address bit.

19.3.2.4 Trace Buffer Transaction Mask Register (TBTMR)

The trace buffer transaction mask register (TBTMR) shown in Figure 19-12 specifies which transaction types to monitor. Each bit in the TBTMR represents a transaction type on the selected interface. The transaction associated with any particular bit depends on the interface being monitored as specified by TBCR1[IFSEL]. Note that the transactions used for defining trace buffer events are the same as those defined for watchpoint monitor events. Thus, Table 19-12 defines the transaction types associated with each interface. Setting a bit enables a hit when this transaction is matched (provided all other match criteria are met and TBCR0[TMD] is clear).

Different interfaces support different transaction types, and the same bit may represent different transaction types depending on the interface.

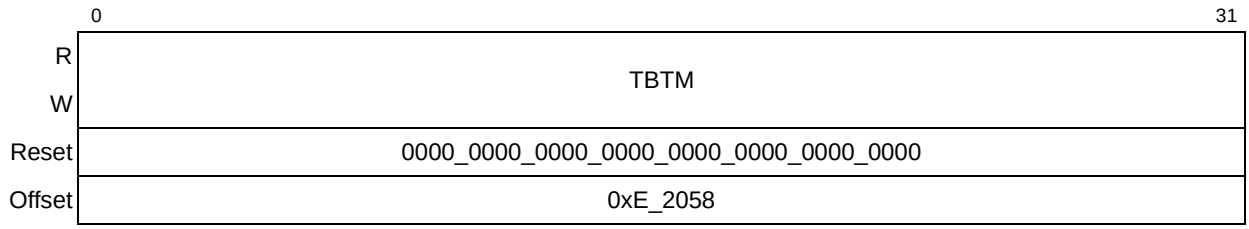


Figure 19-12. Trace Buffer Transaction Mask Register (TBTMR)

Table 19-18 describes the TBTMR field.

Table 19-18. TBTMR Field Descriptions

Bits	Name	Description
0–31	TBTM	Trace buffer transaction mask. Each bit corresponds to a transaction type as defined in Table 19-12. The transaction associated with a bit depends on the interface being monitored. A value of 1 for a given mask bit enables the matching of the transaction associated with that bit. These bits are meaningful only when TBCR0[TMD]=0.

19.3.2.5 Trace Buffer Status Register (TBSR)

The trace buffer status register (TBSR) shown in Figure 19-13 indicates the operational state of the trace buffer.

	0	1	2	3	4																23	24	31
R	ACT	TRIG	STP	WRAP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C_INDX	
W																							
Reset	0000_0000_0000_0000_0000_0000_0000_0000																						
Offset	0xE_205C																						

Figure 19-13. Trace Buffer Status Register (TBSR)

Table 19-19 describes the TBSR fields.

Table 19-19. TBSR Field Descriptions

Bits	Name	Description
0	ACT	Active. Indicates trace buffer activity 0 The start triggering event has not yet occurred. Trace buffer is not armed 1 The start triggering event has occurred. Trace buffer is armed.
1	TRIG	Triggered. Indicates whether or not a programmed event has been triggered 0 The programmed event in TBCR0 has not yet been triggered. 1 The programmed event in TBCR0 has been triggered at least once.
2	STP	Stopped. Indicates whether or not a trace buffer stop condition has been detected. 0 No stop condition yet detected 1 The trace buffer has detected a stop condition and is no longer capturing events.
3	WRAP	Wrapped. Indicates that the trace buffer write pointer has wrapped to the beginning of the buffer at least once. Set when the last entry of the trace buffer is written 0 Pointer has not yet wrapped 1 Pointer has wrapped to the beginning at least once
4–23	—	Reserved
24–31	C_INDX	Current index. Represents the current value of the write pointer at the time TBSR was read. This value may be written by software to initialize the write pointer; however, software is not allowed to write the write pointer while the trace buffer is active. Writes are ignored while the trace buffer is active. It is recommended to write the status register before enabling the trace buffer in order to zero out any bits that might have been set during a prior run and to initialize the write pointer to zero.

19.3.2.6 Trace Buffer Access Control Register (TBACR)

The trace buffer access control register (TBACR), shown in Figure 19-14, enables software to read or write the trace buffer. Each entry is 64 bits; therefore, it takes one write of TBACR and two reads of the access data register (TBADR and TBADHR) to read one 256-entry array entry. Similarly, it takes one write of TBACR and two writes of TBADR and TBADHR to write one array entry. Software can access any entry by writing the appropriate index into TBACR[INDX]. To read or write the buffer sequentially, starting with entry 0, the index must start with a value of 0 and increment every time a new entry is accessed.

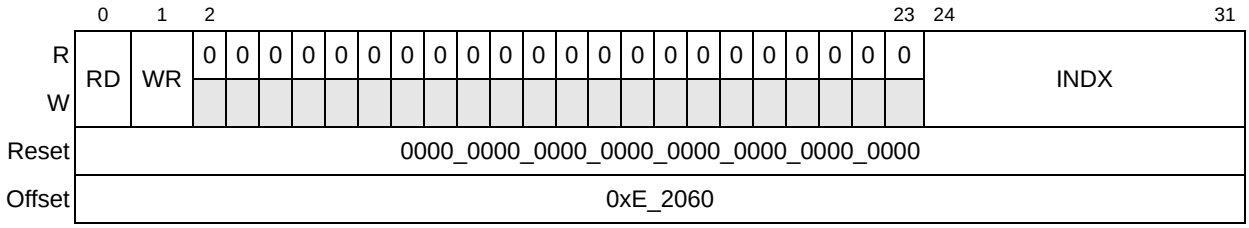


Figure 19-14. Trace Buffer Access Control Register (TBACR)

Table 19-20 describes the TBACR fields.

Table 19-20. TBACR Field Descriptions

Bits	Name	Description
0	RD	Read command. When set, a trace buffer read is performed using the value of TBACR[INDX]. This bit is automatically cleared when the read is performed.
1	WR	Write command. When set, a trace buffer write is performed using the value of TBACR[INDX]. This bit is automatically cleared when the write is performed. A write occurs only if the trace buffer is not active: write requests are ignored while the buffer is active.
2–23	—	Reserved
24–31	INDX	Buffer index to read from or write into (0–255). Used in conjunction with TBACR[RD] and TBACR[WR]

19.3.2.7 Trace Buffer Access Data High Register (TBADHR)

The trace buffer access data high register (TBADHR), shown in Figure 19-15, contains the high-order 32 bits of the data read from the trace buffer during a software-initiated read command (TBACR[RD]), or the write data to be written into the trace buffer during a software-initiated write command (TBACR[WR]). TBACR must be configured to perform a read before this register contains valid data. This register must be initialized by software before configuring the TBACR to perform a write command.

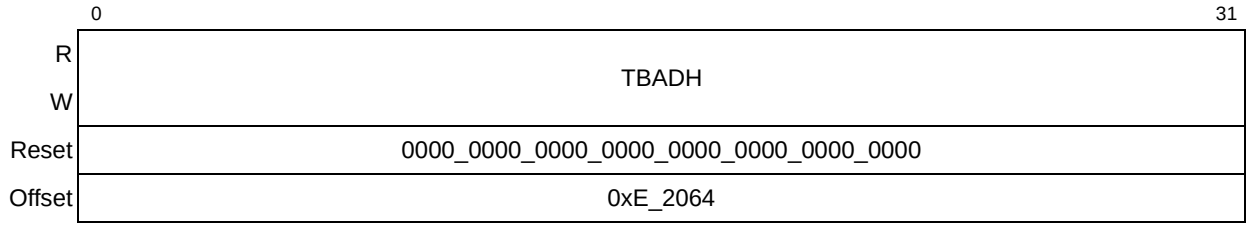


Figure 19-15. Trace Buffer Read High Register (TBADHR)

Table 19-21 describes TBADHR.

Table 19-21. TBADHR Field Descriptions

Bits	Name	Description
0–31	TBADH	Trace buffer access data high. The higher 32 bits of the data read from or to be written into the trace buffer, depending on whether the array is accessed with a read or a write.

19.3.2.8 Trace Buffer Access Data Register (TBADR)

The trace buffer access data register (TBADR), shown in Figure 19-16, contains the low-order 32 bits of the data read from the trace buffer during a software-initiated read command (TBACR[RD]) or the write data to be written into the trace buffer during a software-initiated write command (TBACR[WR]). TBACR must be configured to perform a read before this register contains valid data. This register must be initialized by software before configuring the TBACR to perform a write command.

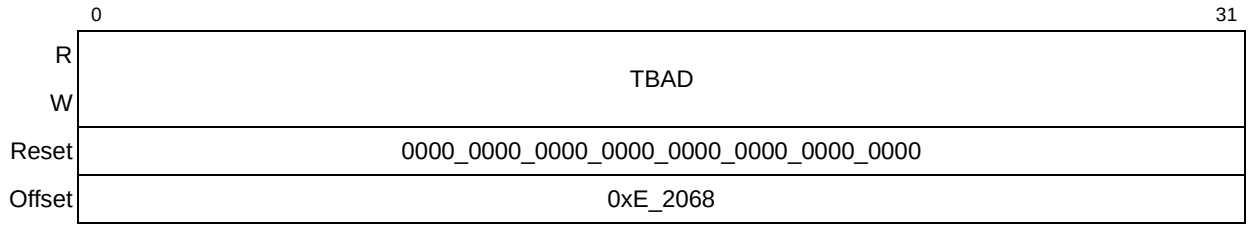


Figure 19-16. Trace Buffer Access Data Register (TBADR)

Table 19-22 describes the TBADR field.

Table 19-22. TBADR Field Descriptions

Bits	Name	Description
0–31	TBAD	Trace buffer access data. Corresponds to the lower 32 bits of the data read from the trace buffer or to be written into the trace buffer, depending on whether software is accessing the array with a read or a write.

19.3.3 Context ID Registers

This section describes the context ID registers. The current context ID register (CCIDR) and programmed context ID registers (PCIDR) are set by software and facilitate debugging complex software.

19.3.3.1 Programmed Context ID Register (PCIDR)

The programmed context ID register (PCIDR), shown in Figure 19-17, contains the user-programmed context ID. This register can be configured to trigger watchpoint events when its value matches the current context ID register (CCIDR), as controlled by WMCR0[ECEN] and WMCR0[NECEN]. See Section 19.3.1.1, "Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1)," for more information.

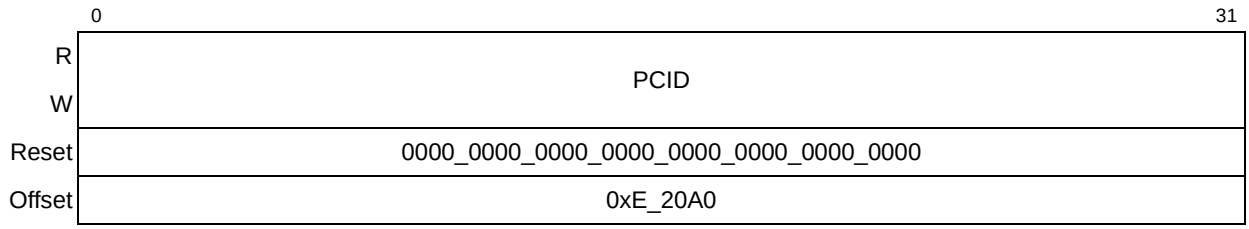


Figure 19-17. Programmed Context ID Register (PCIDR)

Table 19-23 describes the PCIDR field.

Table 19-23. PCIDR Field Descriptions

Bits	Name	Description
0–31	PCID	Programmed context ID. Contains the user-programmed context ID. Compared with current context ID for context-sensitive event triggering.

19.3.3.2 Current Context ID Register (CCIDR)

The current context ID register (CCIDR) shown in Figure 19-18 contains the current context ID. This register is written by software after a context switch and can be used to trigger events when compared with the programmed context ID register (PCIDR).

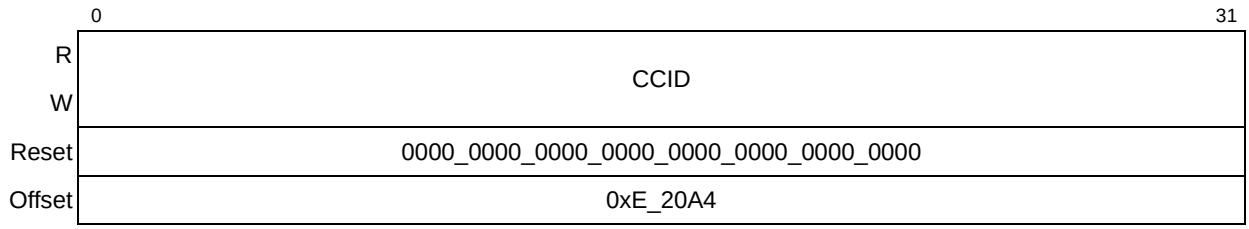


Figure 19-18. Current Context ID Register (CCIDR)

Table 19-24 describes the CCIDR field.

Table 19-24. CCIDR Field Descriptions

Bits	Name	Description
0–31	CCID	Current context ID. Set by user software. Typically loaded immediately following a context switch. Compared with user-programmed context ID for context-sensitive event triggering.

19.3.4 Trigger Out Function

TRIG_OUT provides a convenient mechanism for triggering external system monitors and diagnostic equipment such as logic analyzers. Note that READY is multiplexed with TRIG_OUT. See the last paragraph of Section 4.4.2, “Power-On Reset Sequence,” for more information about READY functionality.

When the trace buffer hit is selected by TOSR[SEL], TRIG_OUT is only meaningful if the trace buffer control register 0 (TBCR0) is properly configured to hit on a traceable event. The same holds true for the watchpoint monitor when the watchpoint monitor is selected by TOSR[SEL].

19.3.4.1 Trigger Out Source Register (TOSR)

The trigger out source register (TOSR) shown in Figure 19-19 specifies the source for TRIG_OUT. The three event-trigger sources are the following:

- The watchpoint monitor
- The trace buffer
- The performance monitor

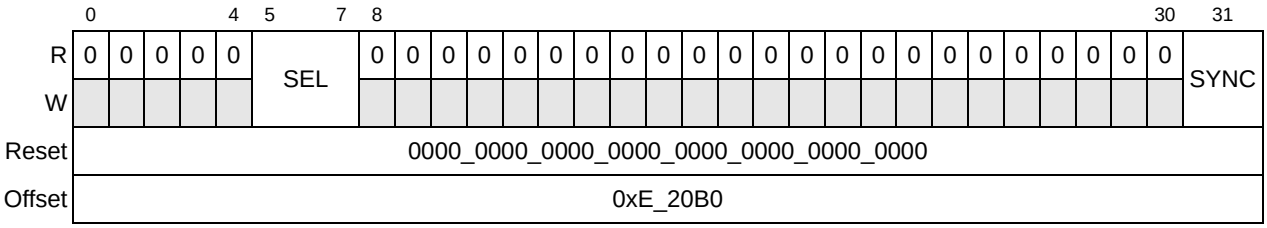


Figure 19-19. Trigger Out Source Register (TOSR)

Table 19-25 describes the TOSR fields.

Table 19-25. TOSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	SEL	Select. Selects the source for TRIG_OUT 000 READY signal. Multiplexed with TRIG_OUT. Basic device state indicator. READY asserts whenever the device is not in reset or not asleep. See Chapter 4, "Reset, Clocking, and Initialization," for more details about the reset sequence, and Chapter 17, "Global Utilities," for more information about power management states. 001 Selects the watchpoint monitor hit indication 010 Selects the trace buffer hit indication 011 Selects the performance monitor overflow indication
8–31	—	Reserved

19.4 Functional Description

The debug features on the MPC8560 use the LBC interfaces, DDR SDRAM interface, and the PCI interface.

19.4.1 Source and Target ID

Debug information that is common to all the interfaces is the source ID (SID). The transaction source ID provides enough information to determine which block or port originated a transaction including the distinction between instruction and data fetches from the processor core. Table 19-26 shows the values and interpretation for the 5-bit SID field. Note that the table also includes ports that are only slaves, such as local memory. These ports are always targets. As such, the value shown represents a target ID (TID) and not a source ID. For ports that can function in both capacities, the value indicates source ID when mastering transactions, and target ID when responding as slave. The TID field is only meaningful when one of the following participates in the transaction:

- The e500 coherency module (ECM) dispatch bus
- The watchpoint monitor (WMCR1[IFSEL] = 000)
- The trace buffer (TBCR1[IFSEL] = 000)

Table 19-26. Source and Target ID Values

Value (Hex)	Source (or Target) Port	Value (Hex)	Source (or Target) Port
00	PCI/PCI-X	10	Local processor (instruction fetch)
01	Reserved	11	Local processor (data fetch)
02	Reserved	12	Reserved

Table 19-26. Source and Target ID Values (continued)

Value (Hex)	Source (or Target) Port	Value (Hex)	Source (or Target) Port
03	Reserved	13	Reserved
04	Local bus controller	14	CPM
05	Reserved	15	DMA
06	Reserved	16	Reserved
07	Reserved	17	System access port (SAP)
08	Configuration space	18	TSEC1
09	Reserved	19	TSEC2
0A	Boot sequencer	1A	Reserved
0B	Reserved	1B	Reserved
0C	RapidIO	1C	RapidIO message unit
0D	Reserved	1D	RapidIO doorbell unit
0E	Reserved	1E	RapidIO port-write unit
0F	Local space (DDR)	1F	Non valid port indicator (reserved for debug info)

19.4.2 PCI/PCI-X Interface Debug

If $\overline{\text{PCI\_GNT3}}$ is low when sampled during POR, the PCI/PCI-X interface operates in debug mode. In debug mode the source ID appears on the high-order address bits (PCI_AD[62:58]) during the bus command phase of a PCI/PCI-X transaction. The bus command phase occurs either during the first cycle that $\overline{\text{PCI\_FRAME}}$ is asserted, or, in the case of addresses greater than 32 bits, after a dual-address cycle phase. In either case, the debug information appears on the highest order address bits while $\overline{\text{PCI\_FRAME}}$ is asserted and both $\overline{\text{PCI\_IRDY}}$ and $\overline{\text{PCI\_TRDY}}$ are negated.

When accessing the low 4 Gbytes of PCI address space for which no dual-address cycle is needed, the debug information appears during the first (and only) address phase on PCI_AD[62:58]. Whenever a dual-address cycle must be run, (addresses above 4 Gbytes) the debug information appears on PCI_AD[62:58] during the second address cycle. In either case a logic analyzer should be configured to sample information on the first cycle of the assertion of $\overline{\text{PCI\_FRAME}}$ and the cycle following a dual-address cycle command.

NOTE

Because they share the same pins, an entire 64-bit address and the debug information cannot be captured in a single cycle.

19.4.3 DDR SDRAM Interface Debug

The DDR interface has two debug modes distinguished by which pins drive the debug information. In one mode, debug information (source ID, data valid) is multiplexed onto the ECC pins; the other mode uses the debug pins.

19.4.3.1 Debug Information on Debug Pins

If MSRCID0 is high when sampled during POR, the debug information from the DDR SDRAM interface is driven on MSRCID[0:4] and MDVAL. This POR value is captured in PORDBGMSR[MEM_SEL] as described in Section 17.4.1.5, "POR Debug Mode Status Register (PORDBGMSR)." In this mode, the source ID appears on MSRCID[0:4] during a RAS or CAS cycle. During any other cycle, the value of MSRCID[0:4] is all ones, which indicates idle cycles on the address/command interface. Similarly, MDVAL is asserted during valid data cycles on the DDR interface.

19.4.3.2 Debug Information on ECC Pins

If MSRCID1 is low when sampled during POR, debug information from the DDR SDRAM interface is selected to appear on MECC[0:5] as shown in Figure 19-1. In this mode, the ID value of the source port, (the source ID), appears on MECC[0:4] during a RAS or CAS cycle. During any other cycle the value of MECC[0:4] is all ones. A data-valid signal (DVAL) is driven on MECC5 during valid DDR SDRAM data cycles.

NOTE

In this mode, MECC[0:5] must be disconnected from all SDRAM devices to prevent contention on those lines.

19.4.4 Local Bus Interface Debug

If MSRCID0 is low when sampled during POR, the LBC is selected as the source for the debug information appearing on MSRCID[0:4] and MDVAL. For more information on this mode, see Section 12.1.3.2, "Source ID Debug Mode."

19.4.5 Watchpoint Monitor

The watchpoint monitor (WM) can be programmed to arm and trigger on many different events including any of the following:

- External event (through TRIG_IN)
- A trace buffer event
- A performance monitor overflow event
- A comparison of the current and programmed context ID registers

A watchpoint event can be used in the following ways:

- Trigger a logic analyzer (using TRIG_OUT)
- Arm or trigger the trace buffer
- Trigger a performance monitor event

The large counters available in the performance monitor block and the interlock between it and the watchpoint monitor support sophisticated debug scenarios.

A WM trigger event may be composed of several events programmed in the watchpoint monitor control registers (WMCR0–WMCR1). Because the watchpoint monitor is disabled by default during POR, these registers must be initialized to make use of this debug feature. Note that the WM address mask register (WMAMR) and the type mask register (WMTMR) are cleared during POR. This means that the watchpoint monitor's default behavior following a power-on reset is to trigger on any address and no transaction type. The reset value of WMCR0[TMD] is 0 which means transaction matching is enabled but since no transaction is selected (WMTMR=0), a match will never occur. Either the transaction matching must be disabled by setting WMCR0[TMD] to a value of 1, or valid transactions must be selected by setting one or more of the WMTMR bits to a value of 1.

19.4.5.1 Watchpoint Monitor Performance Monitor Events

The WM can produce a performance monitor (PM) event with every trigger. This is accomplished by configuring the performance monitor to count WM events. For more information on this configuration see the events named "Number of watchpoint monitor hits," and "Number of trace buffer hits," in Table 18-10.

Multi-level triggers can be created using the watchpoint monitor, the performance monitor, and the trace buffer combined. For example, the WM can be programmed to trigger on events that also increment a PM counter (the performance monitor must also be programmed to respond to this event), the output of which (perfmon_overflow) could trigger the start of tracing in the trace buffer.

19.4.6 Trace Buffer

The trace buffer is a 256×64 array that can capture information about the internal processing of transactions to selected interfaces. The trace buffer controls are a superset of those for the watchpoint monitor. Close inspection of the trace buffer control registers (TBCR*n*) and the WM control registers (WMCR*n*) shows that trace buffer controls not needed for the WM are marked reserved in WMCR*n*. This permits using the trace buffer as a second watchpoint monitor by simply ignoring the trace options.

The trace buffer provides great flexibility about when to start tracing, when to stop tracing, and what to trace. The trace mode field, TBCR0[MODE], indicates when to trace: on every valid cycle, on a watchpoint monitor event, or when all the programmed events in the TBCR are met. This

permits a user to program the trace condition in the watchpoint monitor and to program a start or stop condition in the trace buffer control register. The user can also program the TBCR with the conditions in which to stop tracing: on an event, or when the buffer is full. TBCR0[IFSEL] specifies which interface transactions are being captured.

The trace buffer can be programmed to trace the dispatch bus from any of the following:

- e500 coherency module (ECM)
- Outbound host interface to the RapidIO controller
- Outbound host interface to the PCI controller
- Host interface to the DDR controller

Transactions come into the ECM, arbitrate for common resources, and get dispatched to the target port. Information such as transaction types, source ID, and other attributes can be captured in any of the selected interfaces. The trace buffer can be used to distinguish between requests due to an instruction fetch versus a data fetch on the RapidIO interface, where external I/O alone does not allow for easy discrimination.

19.4.6.1 Traced Data Formats (as a Function of TBCR1[IFSEL])

Figure 19-20 shows the trace buffer entry format for an ECM dispatch (CMD) transaction that is specified when TBCR1[IFSEL] = 000.

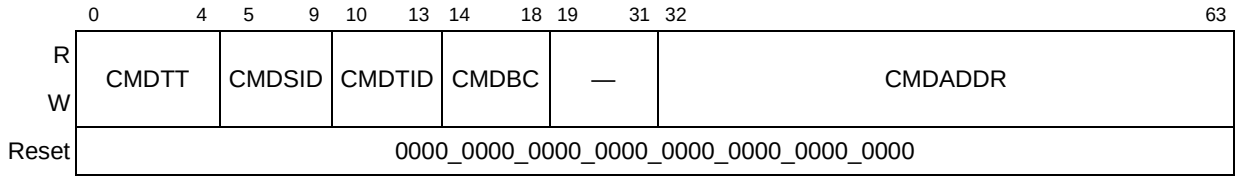


Figure 19-20. e500 Coherency Module Dispatch (CMD) Trace Buffer Entry

Table 19-27 describes the fields of CMD trace buffer entries.

Table 19-27. CMD Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 000)

Bits	Name	Function
0–4	CMDTT	Transaction type. Specifies the transaction type as shown in Table 19-12. For example, a value of zero indicates a write with local processor snoop condition.
5–9	CMDSID	Source ID. Identifies the source of the transaction as shown in Table 19-26. For example, a value of 0b01100 indicates that RapidIO is the transaction source.
10–13	CMDTID	Target ID. Identifies the target of the transaction as shown in Table 19-26. For example, a value of 0b1100 indicates that RapidIO is the transaction target. Note that for a target ID the topmost bit of the 5-bit field is always 0 and, therefore, only 4 bits are required to define a target interface.

Table 19-27. CMD Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 000) (continued)

Bits	Name	Function
14–18	CMDBC	Byte count. Range: 32 to 1 where a value of 0 indicates 32 bytes. 00000 32 bytes 00001 1 byte 00010 2 bytes . . . 11110 30 bytes 11111 31 bytes
19–31	—	Reserved
32–63	CMDADDR	Address bits 0–31

Figure 19-21 shows the trace buffer entry format for the DDR SDRAM interface, TBCR1[IFSEL] = 001.

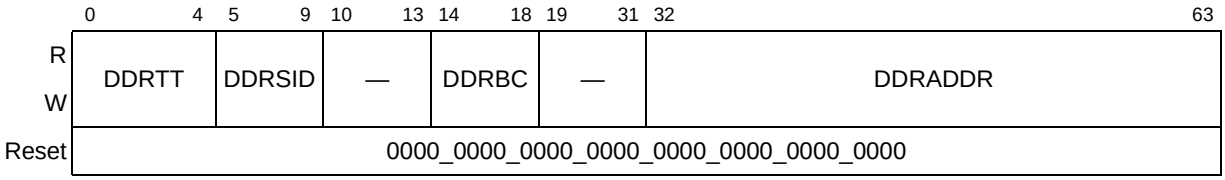


Figure 19-21. DDR Trace Buffer Entry

Table 19-28 describes the fields of DDR SDRAM trace buffer entries when TBCR1[IFSEL] = 001.

Table 19-28. DDR Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 001)

Bits	Name	Function
0–4	DDRTT	Transaction type. Specifies the transaction type as shown in Table 19-12. For example, a value of all zeros maps to write.
5–9	DDRSID	Source ID. Specifies the source of the transaction as shown in Table 19-26. For example, a value of 01100 indicates that RapidIO is the transaction source, and so on.
10–13	—	Reserved
14–18	DDRBC	Byte count
19–31	—	Reserved
32–63	DDRADDR	Address bits 0–31

Figure 19-22 shows the PCI trace buffer entry format when TBCR1[IFSEL] = 010.

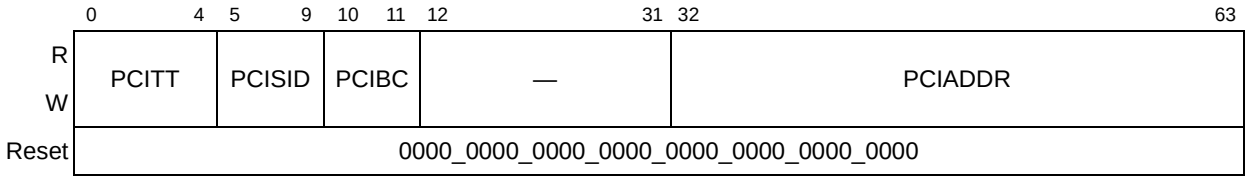


Figure 19-22. PCI Trace Buffer Entry

Table 19-29 describes the fields of PCI trace buffer entries when TBCR1[IFSEL] = 010.

Table 19-29. PCI Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 010)

Bits	Name	Function
0–4	PCITT	Transaction type. Specifies the transaction type as shown in Table 19-12. For example, a value of all zeros maps to write.
5–9	PCISID	Source ID. Identifies the source of the transaction as shown in Table 19-26. For example, a value of 01100 identifies RapidIO as the transaction source.
10–11	PCIBC	Byte count. The size of the transaction. 00 32 bytes 01 8 bytes 10 16 bytes 11 24 bytes
12–31	—	Reserved
32–63	PCIADDR	Address bits 0–31

Figure 19-23 shows the format for RapidIO interface trace buffer entries when TBCR1[IFSEL] = 011.

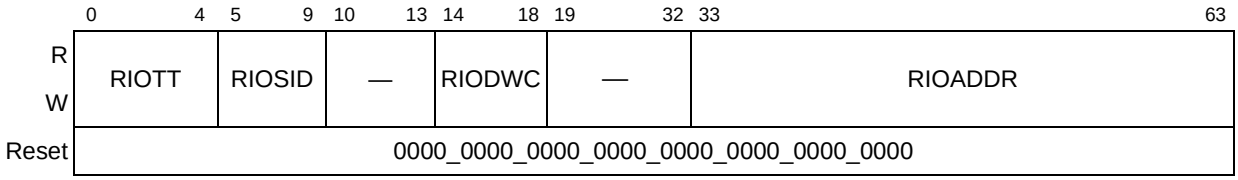


Figure 19-23. RapidIO Trace Buffer Entry

Table 19-30 describes the fields of the RapidIO interface trace buffer entries.

Table 19-30. RapidIO Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 011)

Bits	Name	Function
0–4	RIOTT	Transaction type. Specifies the transaction type as shown in Table 19-12. For example, a value of all zeros indicates an NWRITE transaction.
5–9	RIOSID	Source ID. Identifies the source of the transaction as shown in Table 19-26. For example, a value of 01100 identifies RapidIO as the transaction source.

Table 19-30. RapidIO Trace Buffer Entry Field Descriptions (TBCR1[IFSEL] = 011) (continued)

Bits	Name	Function
10–13	—	Reserved
14–18	RIODWC	Double-word count. This number times eight equals the byte count.
19–32	—	Reserved
33–63	RIOADDR	Address bits 0–30. Represents a double-word address on RapidIO

19.5 Initialization

Configuring the appropriate control register must be the last step in the initialization sequence for either the watchpoint or trace buffer. That is, all required registers except the corresponding control register must be configured before any control register bits that enable watchpoint or trace events are set.

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 MPC8560's four 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 through its serial interface (SIx TDMx and NMSI) ports because the ATM transmission convergence (TC) layer functionality is implemented internally.
- Chapter 39, “Inverse Multiplexing for ATM (IMA),” describes the MPC8560’s implementation of inverse multiplexing for ATM.
- Chapter 40, “CPM Ethernet Controller,” describes the fast Ethernet controller in the CPM.
- Chapter 41, “FCC HDLC Controller,” describes the FCC HDLC controller of the MPC8560.
- Chapter 42, “FCC Transparent Controller,” describes how the FCC transparent controller functions as a high-speed serial-to-parallel and parallel-to-serial converter.
- Chapter 43, “Serial Peripheral Interface (SPI),” describes the serial peripheral interface (SPI) of the MPC8560 CPM.
- Chapter 44, “CPM I^2C Controller,” describes the I^2C controller of the CPM.
- Chapter 45, “Parallel I/O Ports,” describes the four general-purpose I/O ports of the CPM.

Convention

Part V of this document uses the following additional notational convention:

Bold	In figures and tables showing registers and parameter RAM, bold entries indicate fields that should be initialized by the user.

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
  - Freescale Semiconductor 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.

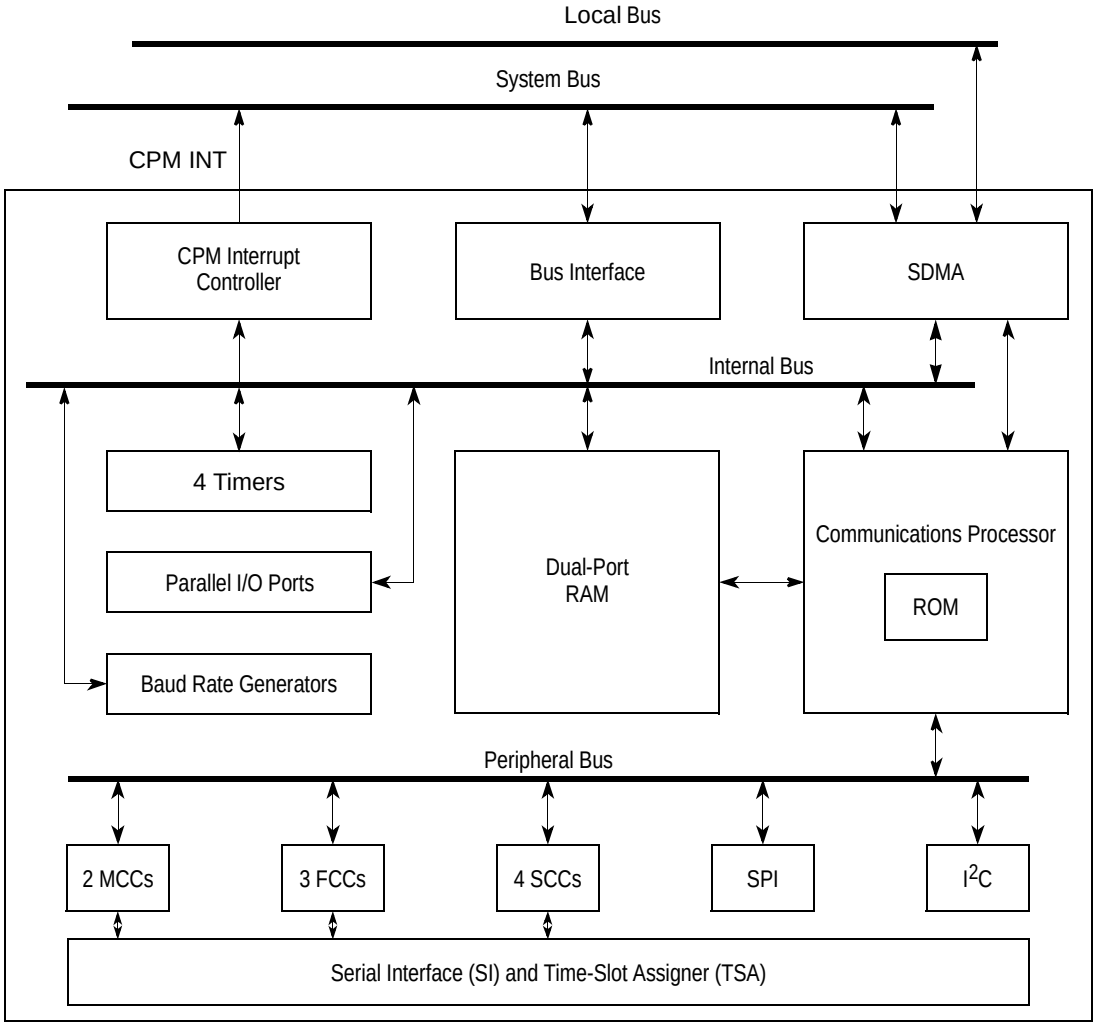


Figure 20-1. MPC8560 CPM Block Diagram

NOTE

The CPM clock frequency is the same as the CCB clock frequency and it is determined by the configuration of the platform PLL during power up reset.

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	DPRAM1—Dual-port RAM	R/W	—	20.5/20-35
0x8_4000–0x8_7FFF	Reserved	—	—	—
0x8_8000–0x8_BFFF	DPRAM2—Dual-port RAM	R/W	—	20.5/20-35
0x8_C000–0x8_FFFF	Reserved	—	—	—
e500 Core Interface				
0x9_0000	CEAR—CPM error address register	R	0x0000_0000	20.3.3.1.1/20-24
0x9_0004	CEER—CPM error event register	R/W	0x0000	20.3.3.1.2/20-25
0x9_0006	CEMR—CPM error mask register	R/W	0x0000	20.3.3.1.3/20-25
SDMA				
0x9_0050	SMAER—System bus address error register	R	0x0000_0000	26.1.1/26-2
0x9_0054	Reserved	—	—	—
0x9_0058	SMEVR—System bus event register	R/W	0x0000_0000	26.1.2/26-2
0x9_005C	SMCTR—System bus control register	R/W	0x3800_0000	26.1.3/26-3
0x9_0060	LMAER—Local bus address error register	R	0x0000_0000	26.1.1/26-2
0x9_0064	Reserved	—	—	—
0x9_0068	LMEVR—Local bus event register	R/W	0x0000_0000	26.1.2/26-2
0x9_006C	LMCTR—Local bus control register	R/W	0x3800_0000	26.1.3/26-3
Interrupt Controller				
0x9_0C00	SICR—CPM interrupt configuration register	R/W	0x0000_0000	21.5.1.1/21-9
0x9_0C02	Reserved	—	—	—
0x9_0C04	SIVEC—CPM interrupt vector register	R/W	0x0000_0000	21.5.1.5/21-14
0x9_0C08	SIPNR_H—CPM interrupt pending register (high)	R/W	0x0000_0000	21.5.1.3/21-11
0x9_0C0C	SIPNR_L—CPM interrupt pending register (low)	R/W	0x0000_0000	21.5.1.3/21-11
0x9_0C10	Reserved	—	—	—
0x9_0C14	SCPRR_H—CPM interrupt priority register (high)	R/W	0x0530_9770	21.5.1.2/21-10

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_0C18	SCPRR_L—CPM interrupt priority register (low)	R/W	0x0530_9770	21.5.1.2/21-10
0x9_0C1C	SIMR_H—CPM interrupt mask register (high)	R/W	0x0000_0000	21.5.1.4/21-13
0x9_0C20	SIMR_L—CPM interrupt mask register (low)	R/W	0x0000_0000	21.5.1.4/21-13
0x9_0C24	SIEXR—CPM external interrupt control register	R/W	0x0000_0000	21.5.1.6/21-15
0x9_0C28–0x9_0C7F	Reserved	—	—	—
Clock				
0x9_0C80	SCCR—System clock control register	R/W	0x0000_0000	24.1/24-2
Input/Output Port				
0x9_0D00	PDIRA—Port A data direction register	R/W	0x0000_0000	45.2.3/45-3
0x9_0D04	PPARA—Port A pin assignment register	R/W	0x0000_0000	45.2.4/45-4
0x9_0D08	PSORA—Port A special options register	R/W	0x0000_0000	45.2.5/45-5
0x9_0D0C	PODRA—Port A open drain register	R/W	0x0000_0000	45.2.1/45-2
0x9_0D10	PDATA—Port A data register	R/W	0x0000_0000	45.2.2/45-2
0x9_0D14–0x9_0D1F	Reserved	—	—	—
0x9_0D20	PDIRB—Port B data direction register	R/W	0x0000_0000	45.2.3/45-3
0x9_0D24	PPARB—Port B pin assignment register	R/W	0x0000_0000	45.2.4/45-4
0x9_0D28	PSORB—Port B special options register	R/W	0x0000_0000	45.2.5/45-5
0x9_0D2C	PODRB—Port B open drain register	R/W	0x0000_0000	45.2.1/45-2
0x9_0D30	PDATB—Port B data register	R/W	0x0000_0000	45.2.2/45-2
0x9_0D34–0x9_0D3F	Reserved	—	—	—
0x9_0D40	PDIRC—Port C data direction register	R/W	0x0000_0000	45.2.3/45-3
0x9_0D44	PPARC—Port C pin assignment register	R/W	0x0000_0000	45.2.4/45-4
0x9_0D48	PSORC—Port C special options register	R/W	0x0000_0000	45.2.5/45-5
0x9_0D4C	PODRC—Port C open drain register	R/W	0x0000_0000	45.2.1/45-2
0x9_0D50	PDATC—Port C data register	R/W	0x0000_0000	45.2.2/45-2
0x9_0D54–0x9_0D5F	Reserved	—	—	—
0x9_0D60	PDIRD—Port D data direction register	R/W	0x0000_0000	45.2.3/45-3
0x9_0D64	PPARD—Port D pin assignment register	R/W	0x0000_0000	45.2.4/45-4
0x9_0D68	PSORD—Port D special options register	R/W	0x0000_0000	45.2.5/45-5

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_0D6C	PODRD—Port D open drain register	R/W	0x0000_0000	45.2.1/45-2
0x9_0D70	PDATD—Port D data register	R/W	0x0000_0000	45.2.2/45-2
CPM Timers				
0x9_0D80	TGCR1—Timer 1 and timer 2 global configuration register	R/W	0x00	25.2.2/25-4
0x9_0D81	Reserved	—	—	—
0x9_0D84	TGCR2—Timer 3 and timer 4 global configuration register	R/W	0x00	25.2.2/25-4
0x9_0D85–0x9_0D8F	Reserved	—	—	—
0x9_0D90	TMR1—Timer 1 mode register	R/W	0x0000	25.2.3/25-6
0x9_0D92	TMR2—Timer 2 mode register	R/W	0x0000	25.2.3/25-6
0x9_0D94	TRR1—Timer 1 reference register	R/W	0x0000	25.2.4/25-7
0x9_0D96	TRR2—Timer 2 reference register	R/W	0x0000	25.2.4/25-7
0x9_0D98	TCR1—Timer 1 capture register	R/W	0x0000	25.2.5/25-8
0x9_0D9A	TCR2—Timer 2 capture register	R/W	0x0000	25.2.5/25-8
0x9_0D9C	TCN1—Timer 1 counter	R/W	0x0000	25.2.6/25-8
0x9_0D9E	TCN2—Timer 2 counter	R/W	0x0000	25.2.6/25-8
0x9_0DA0	TMR3—Timer 3 mode register	R/W	0x0000	25.2.3/25-6
0x9_0DA2	TMR4—Timer 4 mode register	R/W	0x0000	25.2.3/25-6
0x9_0DA4	TRR3—Timer 3 reference register	R/W	0x0000	25.2.4/25-7
0x9_0DA6	TRR4—Timer 4 reference register	R/W	0x0000	25.2.4/25-7
0x9_0DA8	TCR3—Timer 3 capture register	R/W	0x0000	25.2.5/25-8
0x9_0DAA	TCR4—Timer 4 capture register	R/W	0x0000	25.2.5/25-8
0x9_0DAC	TCN3—Timer 3 counter	R/W	0x0000	25.2.6/25-8
0x9_0DAE	TCN4—Timer 4 counter	R/W	0x0000	25.2.6/25-8
0x9_0DB0	TER1—Timer 1 event register	R/W	0x0000	25.2.7/25-8
0x9_0DB2	TER2—Timer 2 event register	R/W	0x0000	25.2.7/25-8
0x9_0DB4	TER3—Timer 3 event register	R/W	0x0000	25.2.7/25-8
0x9_0DB6	TER4—Timer 4 event register	R/W	0x0000	25.2.7/25-8
0x9_0DB8–0x9_101D	Reserved	—	—	—

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
FCC1				
0x9_1300	GFMR1—FCC1 general mode register	R/W	0x0000_0000	34.2/34-3
0x9_1304	FPSMR1—FCC1 protocol-specific mode register	R/W	0x0000_0000	(ATM) 35.13.2/35-89 (Ethernet) 40.18.1/40-21 (HDLC) 41.6/41-8
0x9_1308	FTODR1—FCC1 transmit on demand register	R/W	0x0000	34.6/34-9
0x9_130A	Reserved	—	—	—
0x9_130C	FDSR1—FCC1 data synchronization register	R/W	0x7E7E	34.5/34-8
0x9_130E	Reserved	—	—	—
0x9_1310	FCCE1—FCC1 event register	R/W	0x0000_0000	(ATM) 35.13.3/35-91 (Ethernet) 40.18.2/40-23 (HDLC) 41.9/41-14
0x9_1312	Reserved	—	—	—
0x9_1314	FCCM1—FCC1 mask register	R/W	0x0000_0000	(ATM) 35.13.3/35-91 (Ethernet) 40.18.2/40-23 (HDLC) 41.9/41-14
0x9_1316	Reserved	—	—	—
0x9_1318	FCCS1—FCC1 status register	R	0x00	41.10/41-16 (HDLC)
0x9_1319	Reserved	—	—	—
0x9_131C	FTIRR1_PHY0—FCC1 transmit internal rate registers for PHY0	R/W	0x00	35.13.8/35-96 (ATM)
0x9_131D	FTIRR1_PHY1—FCC1 transmit internal rate registers for PHY1	R/W	0x00	35.13.8/35-96 (ATM)
0x9_131E	FTIRR1_PHY2—FCC1 transmit internal rate registers for PHY2	R/W	0x00	35.13.8/35-96 (ATM)
0x9_131F	FTIRR1_PHY3—FCC1 transmit internal rate registers for PHY3	R/W	0x00	35.13.8/35-96 (ATM)

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
FCC2				
0x9_1320	GFMR2—FCC2 general mode register	R/W	0x0000_0000	34.2/34-3
0x9_1324	FPSMR2—FCC2 protocol-specific mode register	R/W	0x0000_0000	(ATM) 35.13.2/35-89 (Ethernet) 40.18.1/40-21 (HDLC) 41.6/41-8
0x9_1328	FTODR2—FCC2 transmit on-demand register	R/W	0x0000	34.6/34-9
0x9_132A	Reserved	—	—	—
0x9_132C	FDSR2—FCC2 data synchronization register	R/W	0x7E7E	34.5/34-8
0x9_132E	Reserved	—	—	—
0x9_1330	FCCE2—FCC2 event register	R/W	0x0000_0000	(ATM) 35.13.3/35-91 (Ethernet) 40.18.2/40-23 (HDLC) 41.9/41-14
0x9_1332	Reserved	—	—	—
0x9_1334	FCCM2—FCC2 mask register	R/W	0x0000_0000	(ATM) 35.13.3/35-91 (Ethernet) 40.18.2/40-23 (HDLC) 41.9/41-14
0x9_1336	Reserved	—	—	—
0x9_1338	FCCS2—FCC2 status register	R	0x00	41.10/41-16 (HDLC)
0x9_1339	Reserved	—	0x00	—
0x9_133C	FTIRR2_PHY0—FCC2 transmit internal rate registers for PHY0	R/W	0x00	35.13.8/35-96 (ATM)
0x9_133D	FTIRR2_PHY1—FCC2 transmit internal rate registers for PHY1	R/W	0x00	35.13.8/35-96 (ATM)
0x9_133E	FTIRR2_PHY2—FCC2 transmit internal rate registers for PHY2	R/W	0x00	35.13.8/35-96 (ATM)
0x9_133F	FTIRR2_PHY3—FCC2 transmit internal rate registers for PHY3	R/W	0x00	35.13.8/35-96 (ATM)

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
FCC3				
0x9_1340	GFMR3—FCC3 general mode register	R/W	0x0000_0000	34.2/34-3
0x9_1344	FPSMR3—FCC3 protocol-specific mode register	R/W	0x0000_0000	(ATM) 35.13.2/35-89 (Ethernet) 40.18.1/40-21 (HDLC) 41.6/41-8
0x9_1348	FTODR3—FCC3 transmit on-demand register	R/W	0x0000	34.6/34-9
0x9_134A	Reserved	—	—	—
0x9_134C	FDSR3—FCC3 data synchronization register	R/W	0x7E7E	34.5/34-8
0x9_134E	Reserved	—	—	—
0x9_1350	FCCE3—FCC3 event register	R/W	0x0000_0000	(ATM) 35.13.3/35-91 (Ethernet) 40.18.2/40-23 (HDLC) 41.9/41-14
0x9_1352	Reserved	—	—	—
0x9_1354	FCCM3—FCC3 mask register	R/W	0x0000_0000	(ATM) 35.13.3/35-91 (Ethernet) 40.18.2/40-23 (HDLC) 41.9/41-14
0x9_1356	Reserved	—	—	—
0x9_1358	FCCS3—FCC3 status register	R	0x00	41.10/41-16 (HDLC)
0x9_1359–0x9_137F	Reserved	—	—	—
FCC1 (continued)				
0x9_1380	FIRPER1—FCC1 internal rate port enable register	R/W	0x0000_0000	35.13.5/35-92
0x9_1384	FIRER1—FCC1 internal rate event register	R/W	0x0000_0000	35.13.6/35-93
0x9_1388	FIRSR1_HI—FCC1 internal rate selection register:HI	R/W	0x0000_0000	35.13.7/35-94
0x9_138C	FIRSR1_LO— FCC1 internal rate selection register:LO	R/W	0x0000_0000	35.13.7/35-94
0x9_1390	GFEMR1—General FCC1 expansion mode register	R/W	0x00	34.3/34-7
0x9_1391–0x9_139F	Reserved	—	—	—

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
FCC2 (continued)				
0x9_13A0	FIRPER2—FCC2 internal rate port enable register	R/W	0x0000_0000	35.13.5/35-92
0x9_13A4	FIRER2—FCC2 internal rate event register	R/W	0x0000_0000	35.13.6/35-93
0x9_13A8	FIRSR2_HI—FCC2 internal rate selection register:HI	R/W	0x0000_0000	35.13.7/35-94
0x9_13AC	FIRSR2_LO—FCC2 internal rate selection register:LO	R/W	0x0000_0000	35.13.7/35-94
0x9_13B0	GFEMR2—General FCC2 expansion mode register	R/W	0x00	34.3/34-7
0x9_13B1–0x9_13CF	Reserved	—	—	—
FCC3 (continued)				
0x9_13D0	GFEMR3—General FCC3 expansion mode register	R/W	0x00	34.3/34-7
0x9_13D1–0x9_13FF	Reserved	—	—	—
TC Layer 1				
0x9_1400	TCMODE1—TC1 mode register	0x0000	0x0000	38.11.1.1/38-8
0x9_1402	CDSMR1—TC1 cell delineation state machine register	0x0000	0x0000	38.11.1.2/38-10
0x9_1404	TCER1—TC1 event register	0x0000	0x0000	38.11.1.3/38-10
0x9_1406	TC_RCC1—TC1 received cell counter			38.11.3.1/38-14
0x9_1408	TCMR1—TC1 mask register	0x0000	0x0000	38.11.1.4/38-11
0x9_140A	TC_FCC1—TC1 filtered cell counter			38.11.3.6/38-14
0x9_140C	TC_CCC1—TC1 corrected cell counter			38.11.3.4/38-14
0x9_140E	TC_ICC1—TC1 idle cell counter			38.11.3.5/38-14
0x9_1410	TC_TCC1—TC1 transmitted cell counter			38.11.3.2/38-14
0x9_1412	TC_ECC1—TC1 error cell counter			38.11.3.3/38-14
0x9_1414	Reserved	—	—	—
TC Layer 2				
0x9_1420	TCMODE2—TC2 mode register	0x0000	0x0000	38.11.1.1/38-8
0x9_1422	CDSMR2—TC2 cell delineation state machine register	0x0000	0x0000	38.11.1.2/38-10
0x9_1424	TCER2—TC2 event register	0x0000	0x0000	38.11.1.3/38-10
0x9_1426	TC_RCC2—TC2 received cell counter			38.11.3.1/38-14
0x9_1428	TCMR2—TC2 mask register	0x0000	0x0000	38.11.1.4/38-11
0x9_142A	TC_FCC2—TC2 filtered cell counter			38.11.3.6/38-14
0x9_142C	TC_CCC2—TC2 corrected cell counter			38.11.3.4/38-14

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_142E	TC_ICC2—TC2 idle cell counter			38.11.3.5/38-14
0x9_1430	TC_TCC2—TC2 transmitted cell counter			38.11.3.2/38-14
0x9_1432	TC_ECC2—TC2 error cell counter			38.11.3.3/38-14
0x9_1434	Reserved	—	—	—
TC Layer 3				
0x9_1440	TCMODE3—TC3 mode register	0x0000	0x0000	38.11.1.1/38-8
0x9_1442	CDSMR3—TC3 cell delineation state machine register	0x0000	0x0000	38.11.1.2/38-10
0x9_1444	TCER3—TC3 event register	0x0000	0x0000	38.11.1.3/38-10
0x9_1446	TC_RCC3—TC3 received cell counter			38.11.3.1/38-14
0x9_1448	TCMR3—TC3 mask register	0x0000	0x0000	38.11.1.4/38-11
0x9_144A	TC_FCC3—TC3 filtered cell counter			38.11.3.6/38-14
0x9_144C	TC_CCC3—TC3 corrected cell counter			38.11.3.4/38-14
0x9_144E	TC_ICC3—TC3 idle cell counter			38.11.3.5/38-14
0x9_1450	TC_TCC3—TC3 transmitted cell counter			38.11.3.2/38-14
0x9_1452	TC_ECC3—TC3 error cell counter			38.11.3.3/38-14
0x9_1454	Reserved	—	—	—
TC Layer 4				
0x9_1460	TCMODE4—TC4 mode register	0x0000	0x0000	38.11.1.1/38-8
0x9_1462	CDSMR4—TC4 cell delineation state machine register	0x0000	0x0000	38.11.1.2/38-10
0x9_1464	TCER4—TC4 event register	0x0000	0x0000	38.11.1.3/38-10
0x9_1466	TC_RCC4—TC4 received cell counter			38.11.3.1/38-14
0x9_1468	TCMR4—TC4 mask register	0x0000	0x0000	38.11.1.4/38-11
0x9_146A	TC_FCC4—TC4 filtered cell counter			38.11.3.6/38-14
0x9_146C	TC_CCC4—TC4 corrected cell counter			38.11.3.4/38-14
0x9_146E	TC_ICC4—TC4 idle cell counter			38.11.3.5/38-14
0x9_1470	TC_TCC4—TC4 transmitted cell counter			38.11.3.2/38-14
0x9_1472	TC_ECC4—TC4 error cell counter			38.11.3.3/38-14
0x9_1474	Reserved	—	—	—

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
TC Layer 5				
0x9_1480	TCMODE5—TC5 mode register			38.11.1.1/38-8
0x9_1482	CDSMR5—TC5 cell delineation state machine register			38.11.1.2/38-10
0x9_1484	TCER5—TC5 event register			38.11.1.3/38-10
0x9_1486	TC_RCC5—TC5 received cell counter			38.11.3.1/38-14
0x9_1488	TCMR5—TC5 mask register			38.11.1.4/38-11
0x9_148A	TC_FCC5—TC5 filtered cell counter			38.11.3.6/38-14
0x9_148C	TC_CCC5—TC5 corrected cell counter			38.11.3.4/38-14
0x9_148E	TC_ICC5—TC5 idle cell counter			38.11.3.5/38-14
0x9_1490	TC_TCC5—TC5 transmitted cell counter			38.11.3.2/38-14
0x9_1492	TC_ECC5—TC5 error cell counter			38.11.3.3/38-14
0x9_1494	Reserved	—	—	—
TC Layer 6				
0x9_14A0	TCMODE6—TC6 mode register	0x0000	0x0000	38.11.1.1/38-8
0x9_14A2	CDSMR6—TC6 cell delineation state machine register	0x0000	0x0000	38.11.1.2/38-10
0x9_14A4	TCER6—TC6 event register	0x0000	0x0000	38.11.1.3/38-10
0x9_14A6	TC_RCC6—TC6 received cell counter			38.11.3.1/38-14
0x9_14A8	TCMR6—TC6 mask register	0x0000	0x0000	38.11.1.4/38-11
0x9_14AA	TC_FCC6—TC6 filtered cell counter			38.11.3.6/38-14
0x9_14AC	TC_CCC6—TC6 corrected cell counter			38.11.3.4/38-14
0x9_14AE	TC_ICC6—TC6 idle cell counter			38.11.3.5/38-14
0x9_14B0	TC_TCC6—TC6 transmitted cell counter			38.11.3.2/38-14
0x9_14B2	TC_ECC6—TC6 error cell counter			38.11.3.3/38-14
0x9_14B4	Reserved	—	—	—
TC Layer 7				
0x9_14C0	TCMODE7—TC7 mode register	0x0000	0x0000	38.11.1.1/38-8
0x9_14C2	CDSMR7—TC7 cell delineation state machine register	0x0000	0x0000	38.11.1.2/38-10
0x9_14C4	TCER7—TC7 event register	0x0000	0x0000	38.11.1.3/38-10
0x9_14C6	TC_RCC7—TC7 received cell counter			38.11.3.1/38-14
0x9_14C8	TCMR7—TC7 mask register	0x0000	0x0000	38.11.1.4/38-11
0x9_14CA	TC_FCC7—TC7 filtered cell counter			38.11.3.6/38-14

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_14CC	TC_CCC7—TC7 corrected cell counter			38.11.3.4/38-14
0x9_14CE	TC_ICC7—TC7 idle cell counter			38.11.3.5/38-14
0x9_14D0	TC_TCC7—TC7 transmitted cell counter			38.11.3.2/38-14
0x9_14D2	TC_ECC7—TC7 error cell counter			38.11.3.3/38-14
0x9_14D4	Reserved	—	—	—
TC Layer 8				
0x9_14E0	TCMODE8—TC8 mode register	0x0000	0x0000	38.11.1.1/38-8
0x9_14E2	CDSMR8—TC8 cell delineation state machine register	0x0000	0x0000	38.11.1.2/38-10
0x9_14E4	TCER8—TC8 event register	0x0000	0x0000	38.11.1.3/38-10
0x9_14E6	TC_RCC8—TC8 received cell counter			38.11.3.1/38-14
0x9_14E8	TCMR8—TC8 mask register	0x0000	0x0000	38.11.1.4/38-11
0x9_14EA	TC_FCC8—TC8 filtered cell counter			38.11.3.6/38-14
0x9_14EC	TC_CCC8—TC8 corrected cell counter			38.11.3.4/38-14
0x9_14EE	TC_ICC8—TC8 idle cell counter			38.11.3.5/38-14
0x9_14F0	TC_TCC8—TC8 transmitted cell counter			38.11.3.2/38-14
0x9_14F2	TC_ECC8—TC8 error cell counter			38.11.3.3/38-14
0x9_14F4	Reserved	—	—	—
TC Layer—General				
0x9_1500	TCGSR—TC general status register	R/W	0x0000	38.11.2.2/38-13
0x9_1502	TCGER—TC general event register	R/W	0x0000	38.11.2.1/38-12
BRGs 5–8				
0x9_15F0	BRGC5—BRG5 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_15F4	BRGC6—BRG6 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_15F8	BRGC7—BRG7 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_15FC	BRGC8—BRG8 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_1600–0x9_185F	Reserved	—	—	—
I^2C				
0x9_1860	I2MOD—I^2C mode register	R/W	0x00	44.4.1/44-6
0x9_1861	Reserved	—	—	—
0x9_1864	I2ADD—I^2C address register	R/W	0x00	44.4.2/44-7

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1865	Reserved	—	—	—
0x9_1868	II2BRG—I^{2}C BRG register	R/W	0x00	44.4.3/44-8
0x9_1869	Reserved	—	—	—
0x9_186C	I2COM—I^{2}C command register	R/W	0x00	44.4.5/44-9
0x9_186D	Reserved	—	—	—
0x9_1870	I2CER—I^{2}C event register	R/W	0x00	44.4.4/44-8
0x9_1871	Reserved	—	—	—
0x9_1874	II2CMR—^{2}C mask register	R/W	0x00	44.4.4/44-8
0x9_1875–0x9_19BF	Reserved	—	—	—
Communications Processor				
0x9_19C0	CPCR—Communications processor command register	R/W	0x0000_0000	20.4.1/20-30
0x9_19C4	RCCR—CP configuration register	R/W	0x0000_0000	20.3.6/20-28
0x9_19C8–0x9_19D5	Reserved	—	—	—
0x9_19D6	RTER—CP timers event register	R/W	0x0000	20.6.4/20-43
0x9_19DA	RTMR—CP timers mask register	R/W	0x0000	20.6.4/20-43
0x9_19DC	RTSCR—CP time-stamp timer control register	R/W	0x0000	20.3.7/20-29
0x9_19DE	Reserved	—	—	—
0x9_19E0	RTSR—CP time-stamp register	R/W	0x0000_0000	20.3.8/20-30
BRGs 1–4				
0x9_19F0	BRGC1—BRG1 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_19F4	BRGC2—BRG2 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_19F8	BRGC3—BRG3 configuration register	R/W	0x0000_0000	24.2/24-3
0x9_19FC	BRGC4—BRG4 configuration register	R/W	0x0000_0000	24.2/24-3
SCC1				
0x9_1A00	GSMR_L1—SCC1 general mode register	R/W	0x0000_0000	27.2/27-3
0x9_1A04	GSMR_H1—SCC1 general mode register	R/W	0x0000_0000	27.2/27-3

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1A08	PSMR1—SCC1 protocol-specific mode register	R/W	0x0000	27.2.1/27-9 28.16/28-13 (UART) 29.8/29-7 (HDLC) 30.11/30-10 (BISYNC) 31.9/31-9 (Transparent)
0x9_1A0A	Reserved	—	—	—
0x9_1A0C	TODR1—SCC1 transmit-on-demand register	R/W	0x0000	27.2.3/27-10
0x9_1A0E	DSR1—SCC1 data synchronization register	R/W	0x7E7E	27.2.2/27-9
0x9_1A10	SCCE1—SCC1 event register	R/W		28.19/28-20 (UART) 29.11/29-13 (HDLC) 30.14/30-16 (BISYNC) 31.12/31-12 (Transparent)
0x9_1A14	SCCM1—SCC1 mask register	R/W	0x0000	
0x9_1A16	Reserved	—	—	—
0x9_1A17	SCCS1—SCC1 status register	R/W	0x00	28.20/28-22 (UART) 29.12/29-15 (HDLC) 30.15/30-16 (BISYNC) 31.13/31-13 (Transparent)
0x9_1A18– 0x9_1A1F	Reserved	—	—	—
SCC2				
0x9_1A20	GSMR_L2—SCC2 general mode register (low)	R/W	0x0000_0000	27.2/27-3
0x9_1A24	GSMR_H2—SCC2 general mode register (high)	R/W	0x0000_0000	27.2/27-3
0x9_1A28	PSMR2—SCC2 protocol-specific mode register	R/W	0x0000	27.2.1/27-9 28.16/28-13 (UART) 29.8/29-7 (HDLC) 30.11/30-10 (BISYNC) 31.9/31-9 (Transparent)
0x9_1A2A	Reserved	—	—	—

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1A2C	TODR2—SCC2 transmit-on-demand register	R/W	0x0000	27.2.3/27-10
0x9_1A2E	DSR2—SCC2 data synchronization register	R/W	0x7E7E	27.2.2/27-9
0x9_1A30	SCCE2—SCC2 event register	R/W	0x0000_0000	28.19/28-20 (UART) 29.11/29-13 (HDLC) 30.14/30-16 (BISYNC) 31.12/31-12 (Transparent)
0x9_1A34	SCCM2—SCC2 mask register	R/W	0x0000	
0x9_1A36	Reserved	—	—	—
0x9_1A37	SCCS2—SCC2 status register	R/W	0x00	28.20/28-22 (UART) 29.12/29-15 (HDLC) 30.15/30-16 (BISYNC) 31.13/31-13 (Transparent)
0x9_1A38–0x9_1A3F	Reserved	—	—	—
SCC3				
0x9_1A40	GSMR_L3—SCC3 general mode register	R/W	0x0000_0000	27.2/27-3
0x9_1A44	GSMR_H3—SCC3 general mode register	R/W	0x0000_0000	
0x9_1A48	PSMR3—SCC3 protocol-specific mode register	R/W	0x0000	27.2.1/27-9 28.16/28-13 (UART) 29.8/29-7 (HDLC) 30.11/30-10 (BISYNC) 31.9/31-9 (Transparent)
0x9_1A4A	Reserved	—	—	—
0x9_1A4C	TODR3—SCC3 transmit on demand register	R/W	0x0000	27.2.3/27-10
0x9_1A4E	DSR3—SCC3 data synchronization register	R/W	0x7E7E	27.2.2/27-9

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1A50	SCCE3—SCC3 event register	R/W	0x0000_0000	28.19/28-20 (UART) 29.11/29-13 (HDLC) 30.14/30-16 (BISYNC) 31.12/31-12 (Transparent)
0x9_1A54	SCCM3—SCC3 mask register	R/W	0x0000	
0x9_1A56	Reserved	—	—	—
0x9_1A57	SCCS3—SCC3 status register	R/W	0x00	28.20/28-22 (UART) 29.12/29-15 (HDLC) 30.15/30-16 (BISYNC) 31.13/31-13 (Transparent)
0x9_1A58–0x9_1A5F	Reserved	—	—	—
SCC4				
0x9_1A60	GSMR_L4—SCC4 general mode register	R/W	0x0000_0000	27.2/27-3
0x9_1A64	GSMR_H4—SCC4 general mode register	R/W	0x0000_0000	27.2/27-3
0x9_1A68	PSMR4—SCC4 protocol-specific mode register	R/W	0x0000	27.2.1/27-9 28.16/28-13 (UART) 29.8/29-7 (HDLC) 30.11/30-10 (BISYNC) 31.9/31-9 (Transparent)
0x9_1A6A	Reserved	—	—	—
0x9_1A6C	TODR4—SCC4 transmit on-demand register	R/W	0x0000	27.2.3/27-10
0x9_1A6E	DSR4—SCC4 data synchronization register	R/W	0x7E7E	27.2.2/27-9
0x9_1A70	SCCE4—SCC4 event register	R/W	0x0000_0000	28.19/28-20 (UART) 29.11/29-13 (HDLC) 30.14/30-16 (BISYNC) 31.12/31-12 (Transparent)
0x9_1A74	SCCM4—SCC4 mask register	R/W	0x0000	
0x9_1A76	Reserved	—	—	—

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1A77	SCCS4—SCC4 status register	—	—	28.20/28-22 (UART) 29.12/29-15 (HDLC) 30.15/30-16 (BISYNC) 31.13/31-13 (Transparent)
0x9_1A78–0x9_1A7F	Reserved	—	—	—
SPI				
0x9_1AA0	SPMODE—SPI mode register	R/W	0x0000	43.4.1/43-7
0x9_1AA2	Reserved	—	—	—
0x9_1AA6	SPIE—SPI event register	R/W	0x00	43.4.2/43-10
0x9_1AA7	Reserved	—	—	—
0x9_1AAA	SPIM—SPI mask register	R/W	0x00	43.4.2/43-10
0x9_1AAB	Reserved	—	—	—
0x9_1AAD	SPCOM—SPI command register	W	0x00	43.4.3/43-11
0x9_1AA7–0x9_1AFF	Reserved	—	—	—
CPM Mux				
0x9_1B00	CMXSI1CR—CPM mux SI1 clock route register	R/W	0x00	23.4.2/23-12
0x9_1B02	CMXSI2CR—CPM mux SI2 clock route register	R/W	0x00	23.4.3/23-13
0x9_1B03	Reserved	—	—	—
0x9_1B04	CMXFCR—CPM mux FCC clock route register	R/W	0x0000_0000	23.4.4/23-14
0x9_1B08	CMXSCR—CPM mux SCC clock route register	R/W	0x0000_0000	23.4.5/23-16
0x9_1B0C	Reserved	—	—	—
0x9_1B0E	CMXUAR—CPM mux UTOPIA address register	R/W	0x0000	23.4.1/23-7
0x9_1B10–0x9_1B1F	Reserved	—	—	—
SI1 Registers				
0x9_1B20	SI1AMR—SI1 TDMA1 mode register	R/W	0x0000	22.5.2/22-18
0x9_1B22	SI1BMR—SI1 TDMB1 mode register	R/W	0x0000	
0x9_1B24	SI1CMR—SI1 TDMC1 mode register	R/W	0x0000	
0x9_1B26	SI1DMR—SI1 TDMD1 mode register	R/W	0x0000	

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1B28	SI1GMR—SI1 global mode register	R/W	0x00	22.5.1/22-18
0x9_1B29	Reserved	—	—	—
0x9_1B2A	SI1CMDR—SI1 command register	R/W	0x00	22.5.4/22-25
0x9_1B2B	Reserved	—	—	—
0x9_1B2C	SI1STR—SI1 status register	R/W	0x00	22.5.5/22-26
0x9_1B2D	Reserved	—	—	—
0x9_1B2E	SI1RSR—SI1 RAM shadow address register	R/W	0x0000	22.5.3/22-24
MCC1 Registers				
0x9_1B30	MCCE1—MCC1 event register	R/W	0x0000	33.12.1/33-30
0x9_1B32	Reserved	—	—	—
0x9_1B34	MCCM1—MCC1 mask register	R/W	0x0000	33.12.1/33-30
0x9_1B36	Reserved	—	—	—
0x9_1B38	MCCF1—MCC1 configuration register	R/W	0x00	33.10/33-27
0x9_1B39–0x9_1B3F	Reserved	—	—	—
SI2 Registers				
0x9_1B40	SI2AMR—SI2 TDMA2 mode register	R/W	0x0000	22.5.2/22-18
0x9_1B42	SI2BMR—SI2 TDMB2 mode register	R/W	0x0000	22.5.2/22-18
0x9_1B44	SI2CMR—SI2 TDMC2 mode register	R/W	0x0000	22.5.2/22-18
0x9_1B46	SI2DMR—SI2 TDMD2 mode register	R/W	0x0000	22.5.2/22-18
0x9_1B48	SI2GMR—SI2 global mode register	R/W	0x00	22.5.1/22-18
0x9_1B49	Reserved	—	—	—
0x9_1B4A	SI2CMDR—SI2 command register	R/W	0x00	22.5.4/22-25
0x9_1B4B	Reserved	—	—	—
0x9_1B4C	SI2STR—SI2 status register	R/W	0x00	22.5.5/22-26
0x9_1B4D	Reserved	—	—	—
0x9_1B4E	SI2RSR—SI2 RAM shadow address register	R/W	0x0000	22.5.3/22-24
MCC2 Registers				
0x9_1B50	MCCE2—MCC2 event register	R/W	0x0000	33.12.1/33-30
0x9_1B52	Reserved	—	—	—
0x9_1B54	MCCM2—MCC2 mask register	R/W	0x0000	33.12.1/33-30

Table 20-1. MPC8560 Internal Memory Map (continued)

Address (offset)	Register	Access	Reset	Section/Page
0x9_1B56	Reserved	—	—	—
0x9_1B58	MCCF2—MCC2 configuration register	R/W	0x00	33.10/33-27
0x9_1B59–0x9_1FFF	Reserved	—	—	—
SI1 RAM				
0x9_2000–0x9_21FF	SI1TxRAM—SI 1 transmit routing RAM			22.4.3/22-10
0x9_2200–0x9_23FF	Reserved	—	—	—
0x9_2400–0x9_25FF	SI1RxRAM—SI 1 receive routing RAM			22.4.3/22-10
0x9_2600–0x9_27FF	Reserved	—	—	—
SI2 RAM				
0x9_2800–0x9_29FF	SI2TxRAM—SI 2 transmit routing RAM			22.4.3/22-10
0x9_2A00–0x9_2BFF	Reserved	—	—	—
0x9_2C00–0x9_2DFF	SI2RxRAM—SI 2 receive routing RAM			22.4.3/22-10
0x9_2E00–0x9_3FFF	Reserved	—	—	—
Instruction RAM				
0xA_0000–0xA_7FFF	Dual-port RAM (instruction RAM only)		undefined	20.5/20-35
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	—	—	—
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.

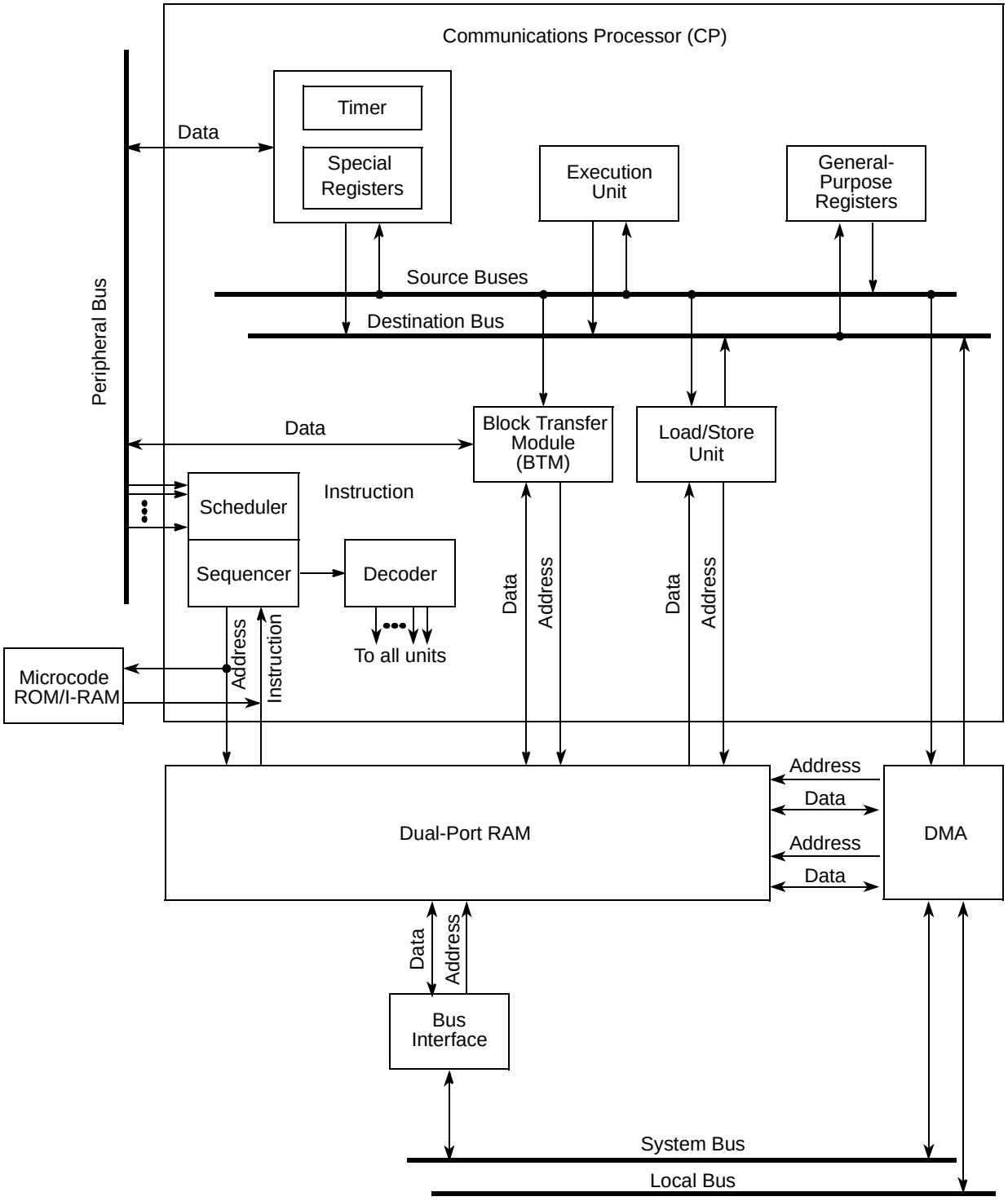


Figure 20-2. Communications Processor (CP) Block Diagram

20.3.3.1 Error Reporting and Capture

When the e500 core (or any other master) tries to read from or write to the CPM registers or internal memories and an erroneous condition is recognized, the CPM reports the event and captures the information in the CPM error registers. The following sections describe these registers.

20.3.3.1.1 CPM Error Address Register (CEAR)

The CPM error address register (CEAR), shown in Figure 20-3, is a read-only register that contains the address and attributes of the first erroneous transaction that occurred after the last time that the CPM error event register (CEER) was entirely cleared. Note that this register contains invalid data if the CPM error event register has all of its bits cleared. The CEER is cleared by $\overline{\text{HRESET}}$, or as a result of a write transaction.

CEAR is not affected by write transactions.

	0	7	8	9	10	15
Field	BYTE_EN		RW	—	ADDR	
R/W	R					
Reset	0000_0000_0000_0000[1]					
Addr	0x90000					

	16	31
Field	ADDR	
R/W	R	
Reset	0000_0000_0000_0000 [1]	
Addr	0x90002	

[1]CEAR is unaffected by soft reset. It is cleared by hard reset.

Figure 20-3. CPM Error Address Register (CEAR)

Table 20-3. CEAR Field Descriptions

Bits	Name	Description
0–7	BYTE_EN	Byte enables. Contains the byte enable pattern of the erroneous transaction.
8	RW	Read/Write. The read/write attribute of the erroneous transaction. 1 Erroneous transaction type is read. 0 Erroneous transaction type is write.
9	—	Reserved
10–31	ADDR	Address. The address of the erroneous transaction. ADDR[29–31] are always read as zeros. This field contains the internal CPM offset, while the full address is a combination of this field and the base address of the CPM as indicated by the CCSRBAR setting.

20.3.3.1.2 CPM Error Event Register (CEER)

The CPM error event register contains the type of error if an error event has occurred. When an erroneous transaction is recognized, the CPM sets the corresponding bit in CEER. Interrupt will be generated if enabled by the CPM error mask register. Note that the CEER will set any bit that corresponds to an erroneous event, and not only the first to occur.

CEER bits are cleared by $\overline{\text{HRESET}}$ or by writing 1 to the appropriate bit; writing 0 has no effect.

	0	1	2	3	4	15
Field	ABORT	BE	SIZE	NOMAP	—	
Reset	0000_0000_0000_0000[1]					
R/W	R/W					
Addr	0x90004					

1 CEER is unaffected by soft reset. It is cleared by hard reset.

Figure 20-4. CPM Error Event Register (CEER)

Table 20-4. CEER Field Descriptions

Bits	Name	Description
0	ABORT	Abort error. Asserted if a transaction that was targeted to the CPM was aborted before completion due to an internal error condition in the MPC8560.
1	BE	Byte enable error. Set when a transaction with a non-contiguous or an all-zeros byte enable pattern is targeted to the CPM.
2	SIZE	SIZE error. Set when an access is attempted to CPM registers, with byte enable pattern that crosses an aligned 4-byte boundary.
3	NOMAP	Not mapped. Set when an access is targeted into a non-implemented address space. Non-implemented space is defined by: CPM Base + 0x14000 ≤ Address ≤ CPM Base + 0x1FFFF, or CPM Base + 0x30000 ≤ Address ≤ CPM Base + 0x3FFFF
4–15	—	Reserved, should be cleared.

20.3.3.1.3 CPM Error Mask Register (CEMR)

The CPM error mask register (CEMR) is used to mask corresponding event bits in CEER. Setting a mask bit enables the corresponding CEER interrupt.

	0	1	2	3	4	15
Field	ABORT	BE	SIZE	NOMAP	—	
Reset	0000_0000_0000_0000					
R/W	R/W					
Addr	0x90006					

Figure 20-5. CPM Error Mask Register (CEMR)

Table 20-5. CEMR Field Descriptions

Bits	Name	Description
0	ABORT	Mask interrupt for ABORT event 0 Interrupt is disabled. 1 Interrupt is enabled.
1	BE	Mask interrupt for BE error event 0 Interrupt is disabled. 1 Interrupt is enabled.
2	SIZE	Mask interrupt for SIZE error event 0 Interrupt is disabled. 1 Interrupt is enabled.
3	NOMAP	Mask interrupt for NOMAP error event 0 Interrupt is disabled. 1 Interrupt is enabled.
4–15	—	Reserved, should be cleared.

20.3.4 Peripheral Interface

The CP uses the peripheral bus to communicate with all of its peripherals. Each FCC and each SCC has 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 20-6 shows the order in which the CP handles requests from peripherals from highest to lowest priority.

Table 20-6. 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

Table 20-6. Peripheral Prioritization (continued)

Priority	Request
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
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 Freescale Semiconductor 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 Freescale and load it into the instruction RAM.

20.3.6 RISC Controller Configuration Register (RCCR)

The RISC controller configuration register (RCCR), shown in Figure 20-6, configures the CP to run microcode from ROM or instruction RAM and controls the CP's internal timer. The register is cleared at reset.

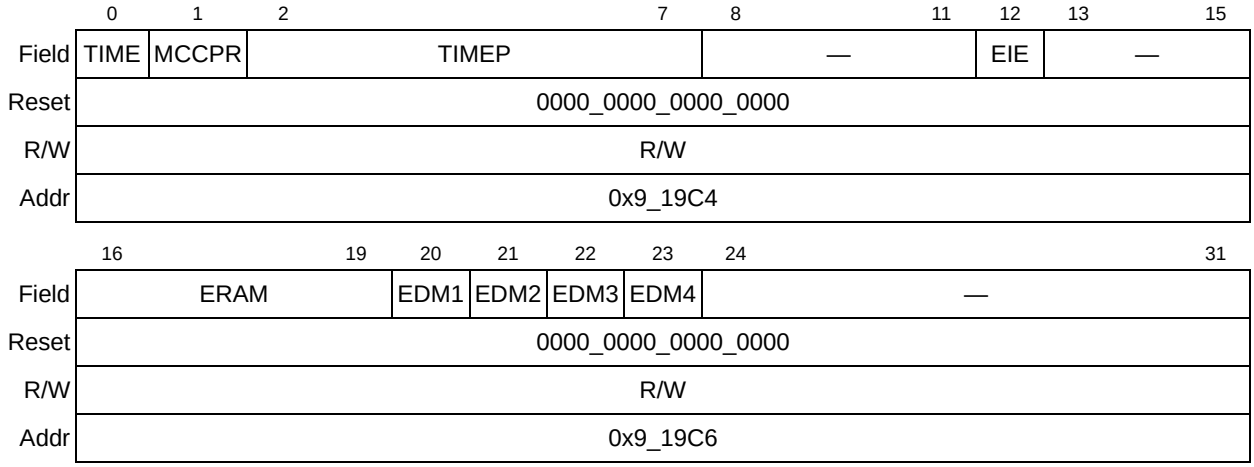


Figure 20-6. RISC Controller Configuration Register (RCCR)

RCCR bit fields are described in Table 20-7.

Table 20-7. 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 20-6 for more information. 0 Original CPM priority scheme. MCC*x* priority behaves according to Table 20-6. 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.

Table 20-7. RISC Controller Configuration Register Field Descriptions (continued)

Bits	Name	Description
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 Freescale-supplied RAM microcode package. 0 DREQ1 cannot interrupt the CP. 1 DREQ1 will interrupt the CP.
13–15	—	Reserved, should be cleared.
16–19	ERAM	Enable RAM microcode. Configure as instructed in the download process of a Freescale-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 20-7, 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 41.6, “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 20-7. RISC Time-Stamp Control Register (RTSCR)

Table 20-8 describes RTSCR fields.

Table 20-8. 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 prescale. 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 20-8, contains the time stamp.

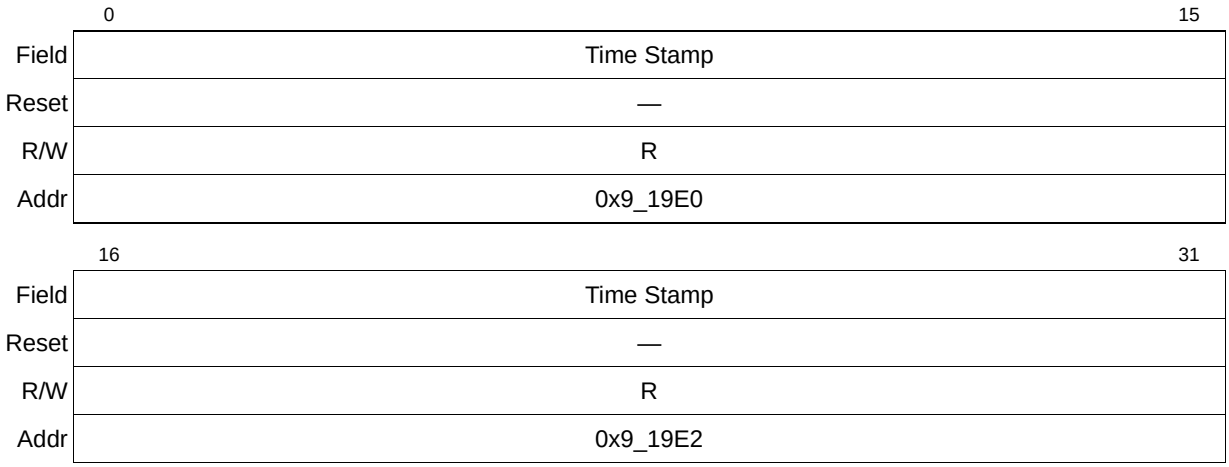


Figure 20-8. RISC Time-Stamp Register (RTSR)

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 a dual-port RAM location called REV_NUM that resides in the miscellaneous parameter RAM. The other locations are reserved for future use.

Table 20-9. 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 CPCR.

20.4.1 CP Command Register (CPCR)

The core should set CPCR[FLG], shown in Figure 20-9, 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.

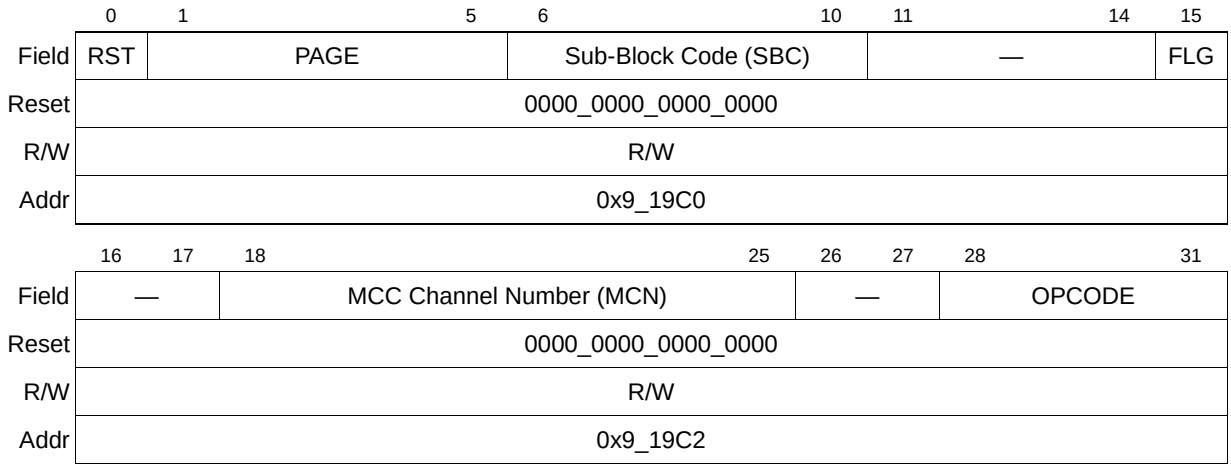


Figure 20-9. CP Command Register (CPCR)

Table 20-10 describes CPCR fields.

Table 20-10. CP Command Register Field Descriptions

Bits	Name	Description
0	RST	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, I^2C, MCC) as well as the CP and RISC timer tables. However, this command does not affect the serial interface (SI*x*) or parallel I/O registers.
1–5	PAGE	Indicates the parameter RAM page number associated with the sub-block being served. See the SBC description for page numbers.

Table 20-10. CP Command Register Field Descriptions (continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></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 (bits 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>I^2C</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 20-11.</td></tr>
</table>

[1] Set according to OPCODE (bits 28–31). If OPCODE is 1010, SBC must be 01110. Refer to Table 20-11.

20.4.1.1 CP Commands

The CP command opcodes are shown in Table 20-11.

Table 20-11. 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 Freescale-supplied RAM microcodes.						

[1] See FCC1 and FCC2 in SBC[6–10] in Table 20-10.

NOTE

If a reserved command is issued, the CPM enters an unknown state that requires an external reset to recover.

The commands in Table 20-11 are described in Table 20-12.

Table 20-12. 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 33.11, “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 33.11, “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 33.11, “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 33.11, “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 33.11, “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 33.11, “MCC Commands.”
MCC STOP RECEIVE	See Section 33.11, “MCC Commands.”

Table 20-12. Command Descriptions (continued)

Command	Description
MCC RESET [1]	MCC reset. Provides a hard reset to the MCC FIFOs. See Section 33.11, “MCC Commands.” To use this command, software should execute the following sequence: 1. Disable the TDM by clearing the appropriate enable bit in SI*x*GMR[4–7]. (See Table 22-4.) 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 SI*x*GMR[4–7]. (See Table 22-4.)
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 20-14

[1] Not available on pre-revision B.3 silicon.

NOTE

The CPM accesses BDs by initiating a DMA cycle on either the system or local bus. If BDs are located in DPRAM, the CPM initiates a cycle on the system bus because DPRAM is a slave device on the system bus. Therefore, a system design should not plan to access the system 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 Kbytes 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 20-10 is a block diagram of the internal RAM.

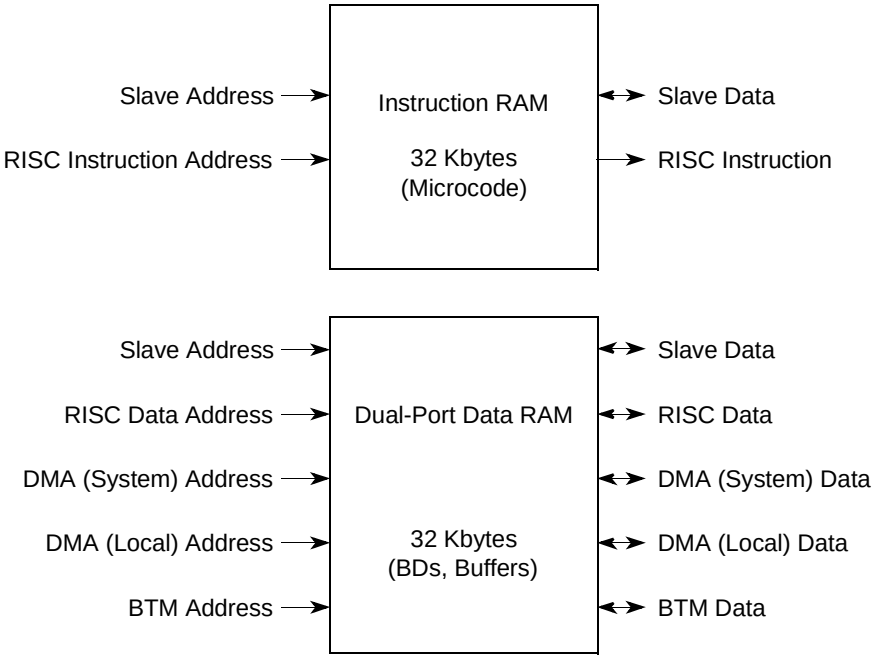


Figure 20-10. Internal RAM Block Diagram

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 20-11 shows the memory map of the internal instruction RAM.

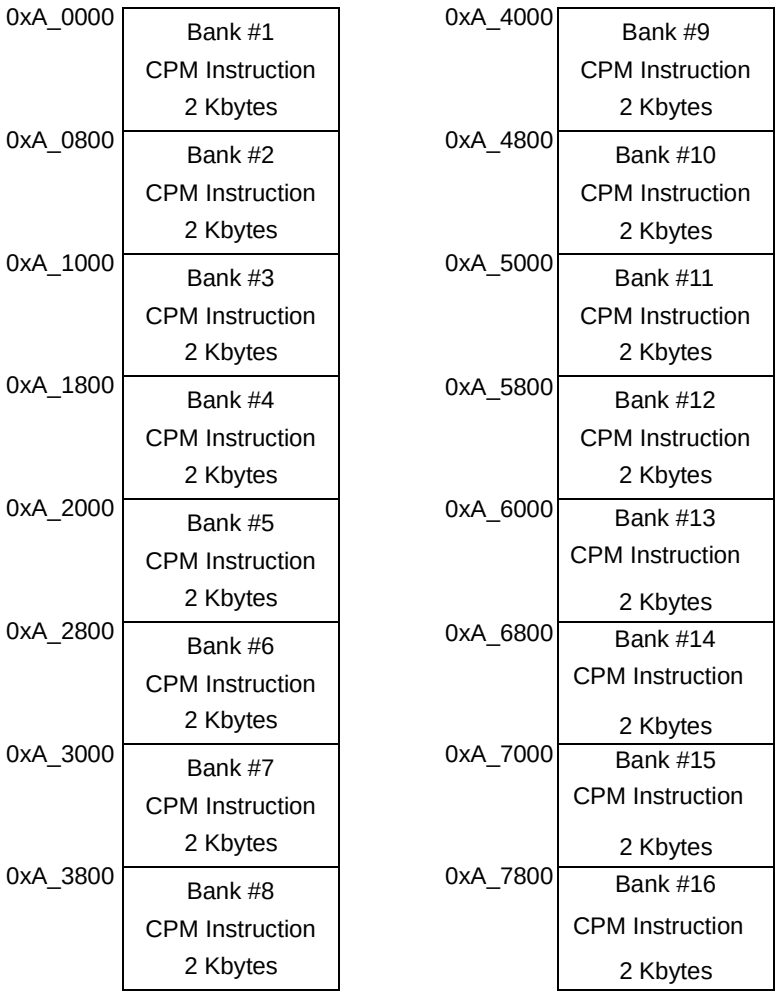


Figure 20-11. Internal Instruction RAM Memory Map

Figure 20-12 shows a memory map of the internal dual-port data RAM.

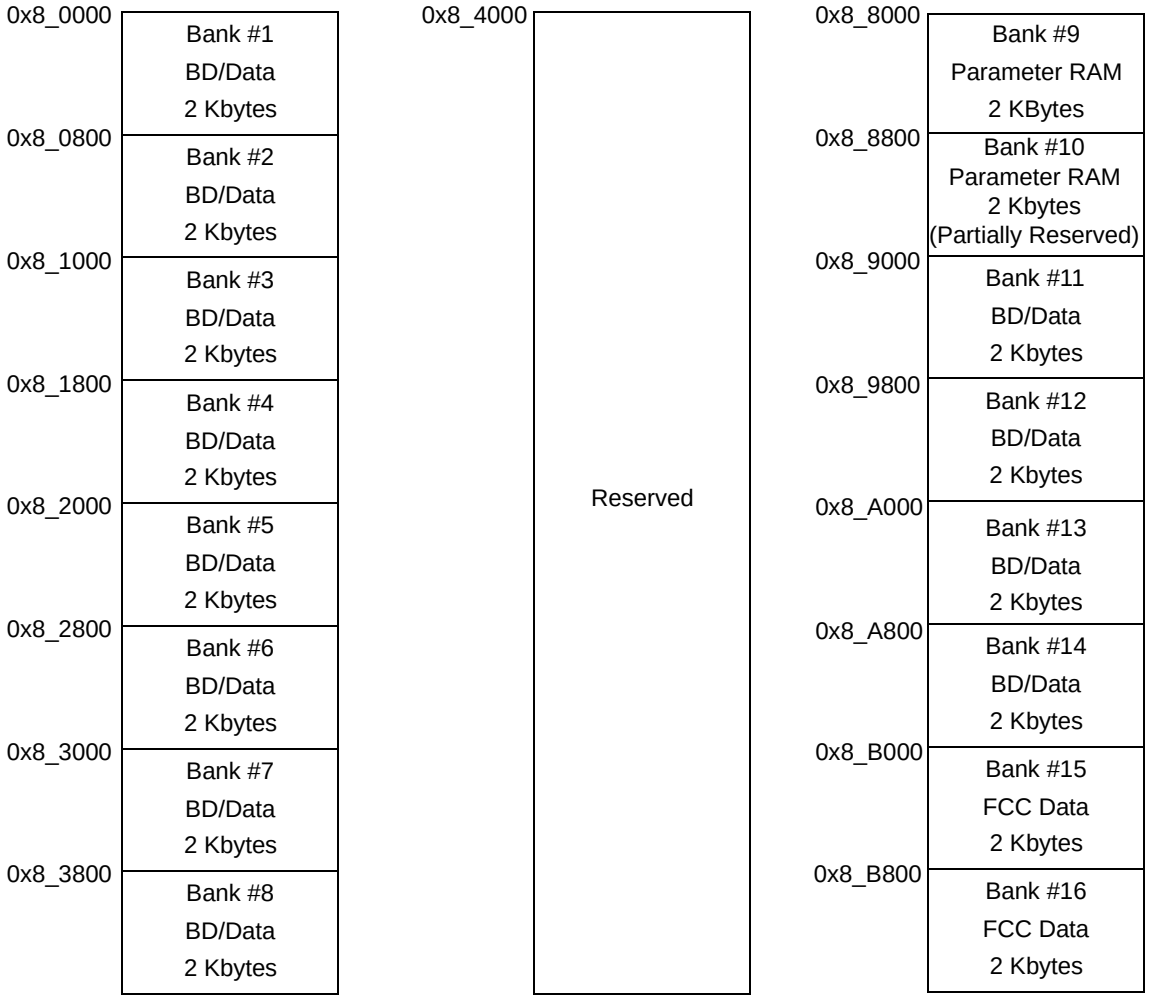


Figure 20-12. Internal Dual-Port Data RAM Memory Map

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 20-13.

Table 20-13. 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 20-14.

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 20-14. 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

Table 20-14. Parameter RAM (continued)

Page	Address Offset	Peripheral	Size (Bytes)
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 20-9.

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 20-13, are used for the RISC timer tables:

- The RISC timer table parameter RAM
- The RISC timer table entries

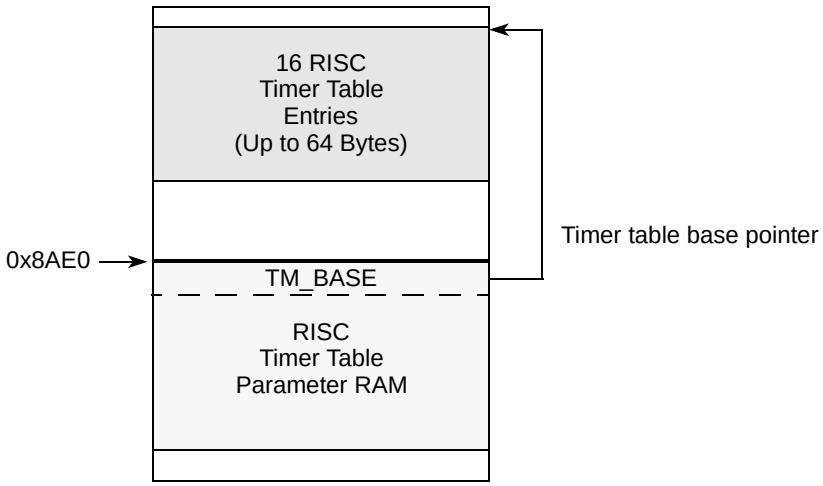


Figure 20-13. RISC Timer Table RAM Usage

The RISC timer table parameter RAM area begins at the RISC timer base address and is used for the general timer parameters; see Table 20-15.

Table 20-15. RISC Timer Table Parameter RAM

Offset[1]	Name[2]	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 completes 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).

2 **Boldfaced** entries must be initialized by the user.

20.6.2 RISC Timer Command Register (TM_CMD)

Figure 20-14 shows the RISC timer command register (TM_CMD).

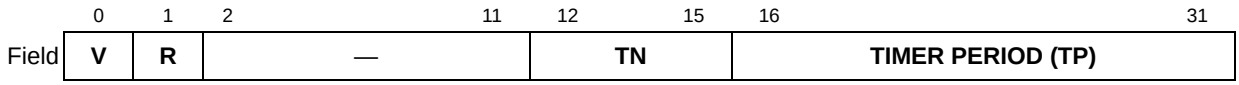


Figure 20-14. RISC Timer Command Register (TM_CMD)

TM_CMD fields are described in Table 20-16.

Table 20-16. TM_CMD Field Descriptions

Bits	Name[1]	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.

[1] **Boldfaced** entries must be initialized by the user.

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 time table bit is set in the CPM interrupt mask register. See Section 21.5.1.4, “CPM Interrupt Mask Registers (SIMR_H and SIMR_L).”

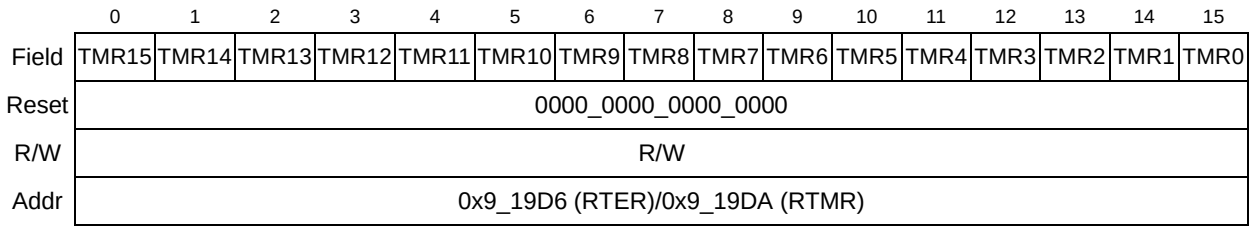


Figure 20-15. RISC Timer Event Register (RTER)/Mask Register (RTMR)

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 0x29E1_0008 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 CPM interrupt mask register (SIMR_L[RTT]) to generate interrupts to the system. The CPM 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 CPM interrupt mask register (SIMR_L) to allow the RISC timers to generate a system interrupt. Initialize the CPM 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 CPM 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 $1024 \times 16 = 16{,}384$ 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 programmable interrupt controller (PIC).

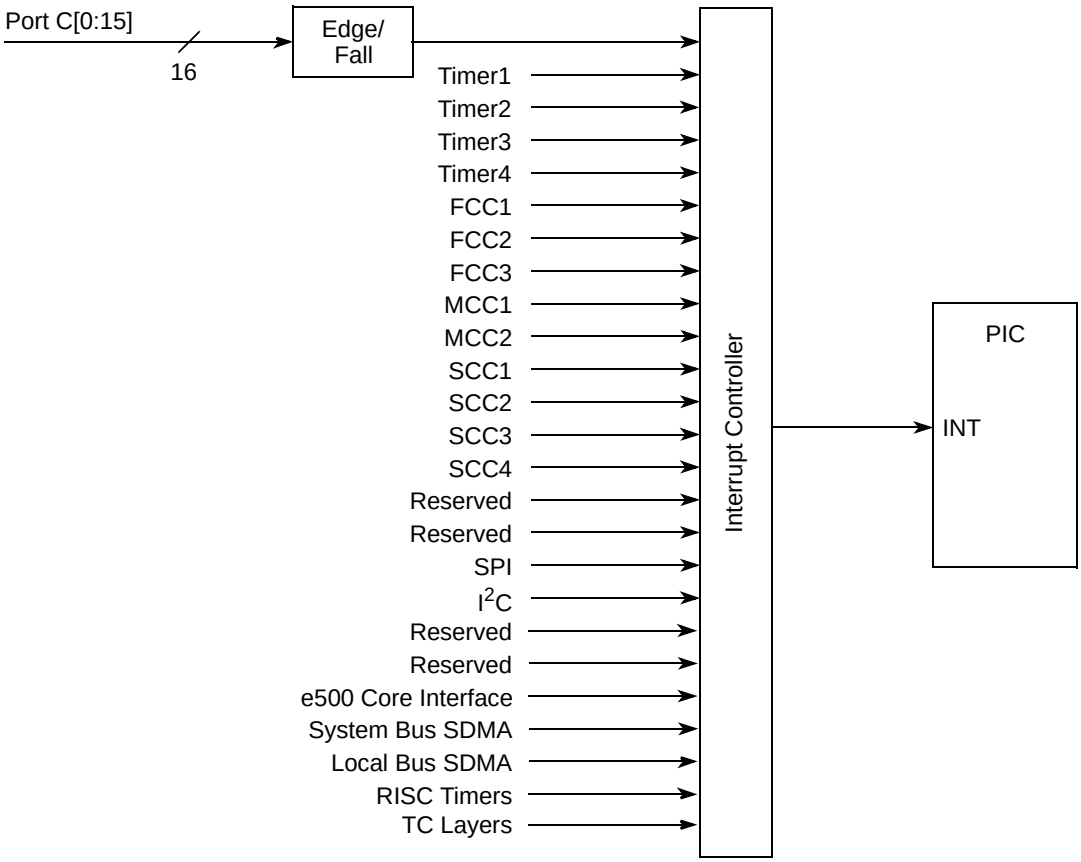


Figure 21-1. MPC8560 CPM Interrupt Structure

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 PIC.

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 PIC. Table 21-1 shows prioritization of all interrupt sources. As described in the 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	Reserved	—
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	I^2C	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.

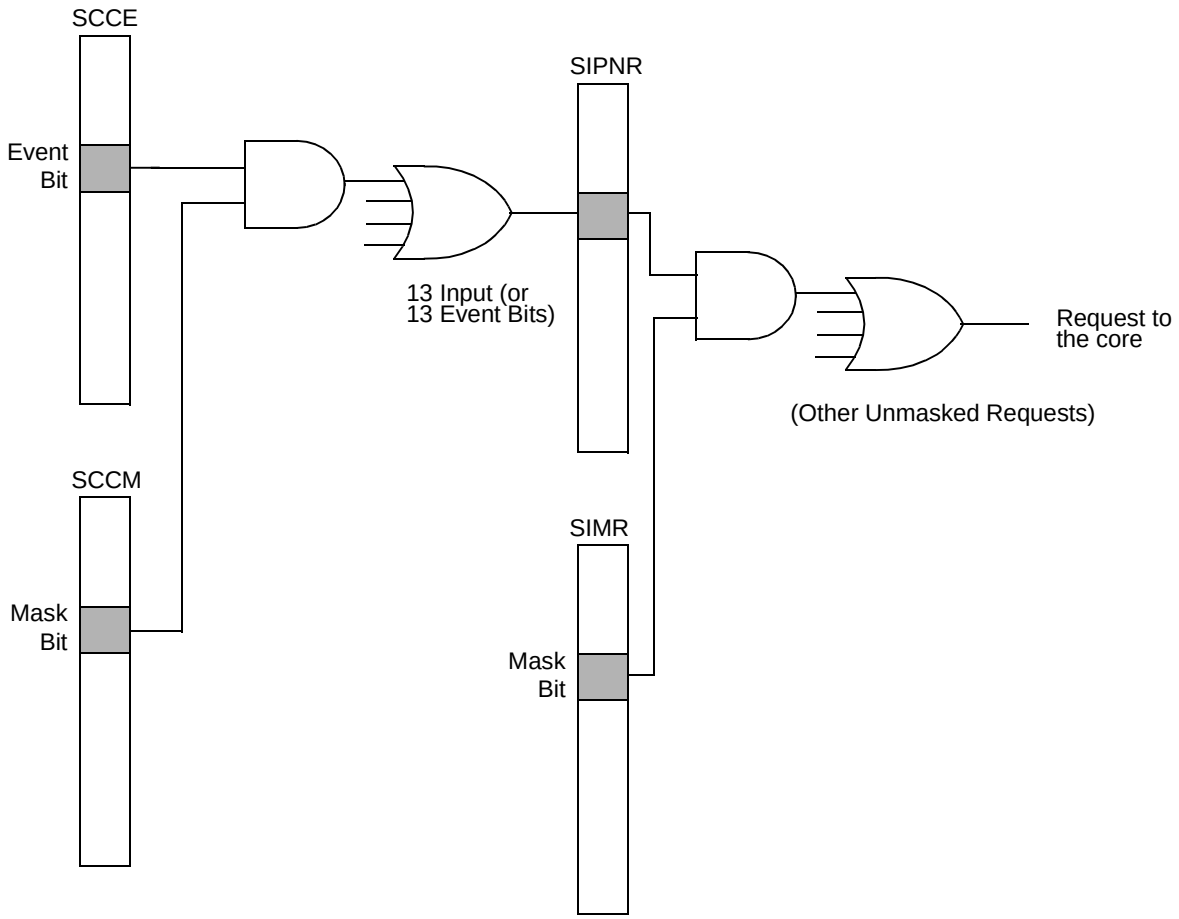


Figure 21-2. Interrupt Request Masking

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
3	RISC timers	0b00_0011

Table 21-2. Encoding the Interrupt Vector (continued)

Interrupt Number	Interrupt Source Description	Interrupt Vector
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	Reserved	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	Core interface	0b10_1101
46	SDMA system	0b10_1110
47	SDMA local	0b10_1111
48	PC15	0b11_0000
49	PC14	0b11_0001

Table 21-2. Encoding the Interrupt Vector (continued)

Interrupt Number	Interrupt Source Description	Interrupt Vector
50	PC13	0b11_0010
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.

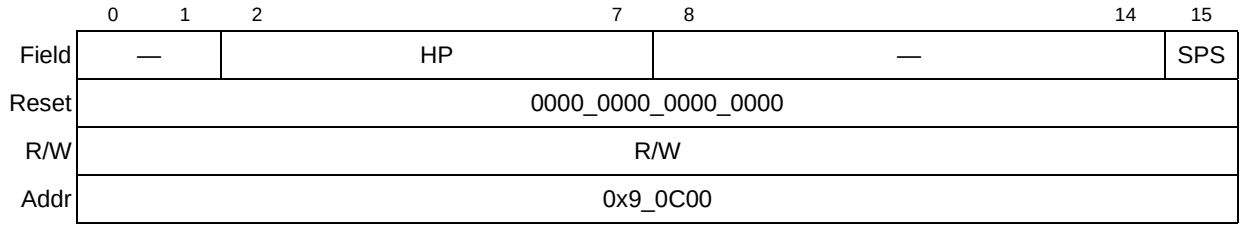


Figure 21-3. CPM Interrupt Configuration Register (SICR)

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, defines priorities between the FCCs and MCCs.

	0–2	3–5	6–8	9–11	12–15
Field	XC1P	XC2P	XC3P	XC4P	—
Reset	000	001	010	011	0000
R/W	R/W				
Addr	0x9_0C14				

	16–18	19–21	22–24	25–27	28–31
Field	XC5P	XC6P	XC7P	XC8P	—
Reset	100	101	110	111	0000
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	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.
3–11, 16–27	XC2P–XC8P	Same as XC1P, but for XCC2–XCC8
12–15, 28–31	—	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_0C1A				

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	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. 101 Core interface asserts its request in the YCC1 position. 110 System SDMA asserts its request in the YCC1 position. 111 Local SDMA asserts its request in the YCC1 position.
3–11, 16–27	YC2P–YC8P	Same as YC1P, but for YCC2–YCC8
12–15, 28-31	—	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	14	15
Field	FCC1	FCC2	FCC3	—	MCC1	MCC2	—		SCC1	SCC2	SCC3	SCC4	TC	Core I/F	SDMA SYS	SDMA LCL
Reset	0000_0000_0000_0000[1]															
R/W	R/W															
Addr	0x9_0C0C															

	16	17	18	19	26	27	28	29	30	31
Field	I^2C	SPI	RTT	—		TIMER1	TIMER2	TIMER3	TIMER4	—
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 the SIMR_L register.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	FCC1	FCC2	FCC3	—	MCC1	MCC2	—		SCC1	SCC2	SCC3	SCC4	TC	Core I/F	SDMA SYS	SDMA LCL
Reset	0000_0000_0000_0000															
R/W	R/W															
Addr	0x9_0C20															

	16	17	18	19	26	27	28	29	30	31
Field	I^2C	SPI	RTT	—		TIMER1	TIMER2	TIMER3	TIMER4	—
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.

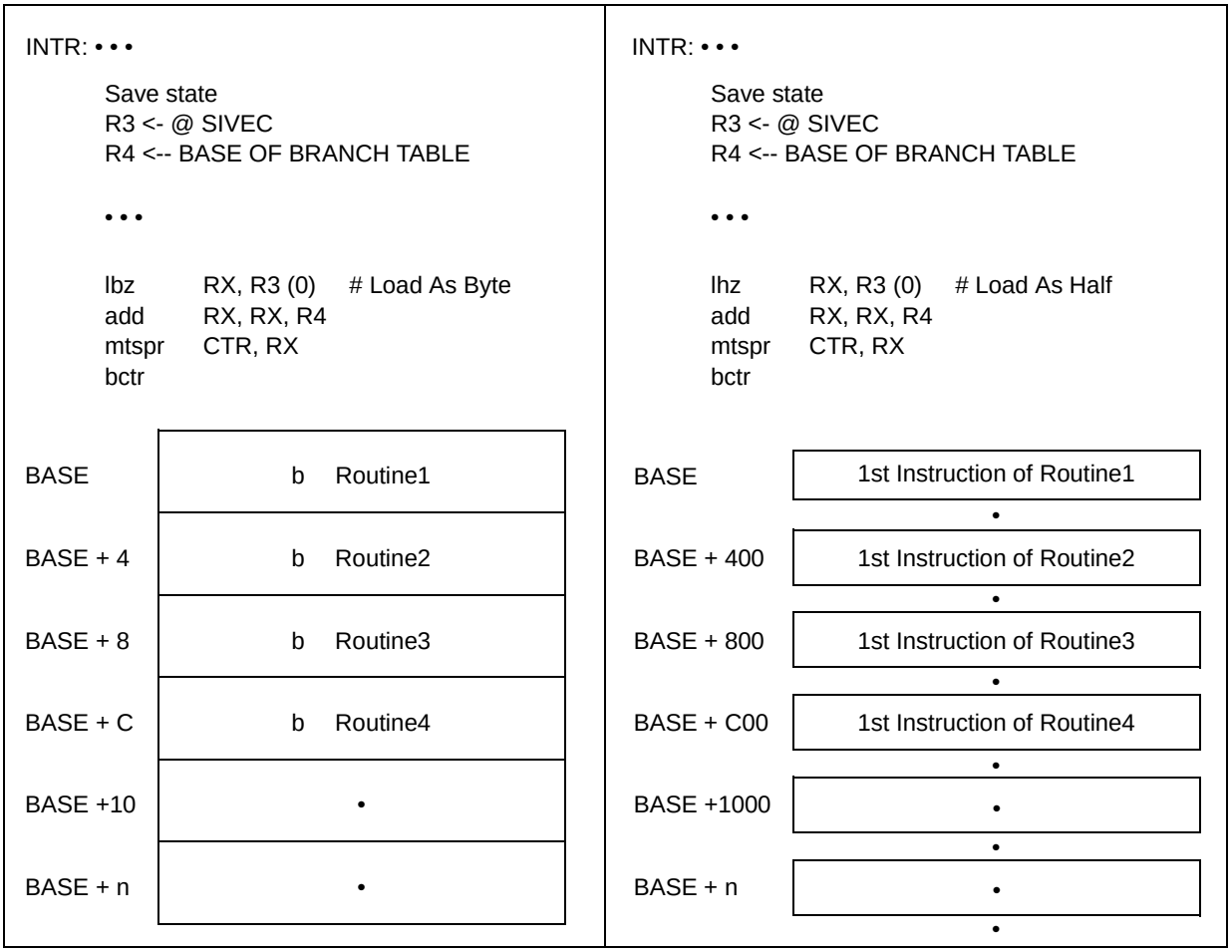


Figure 21-11. Interrupt Table Handling Example

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	EDPC0	EDPC1	EDPC2	EDPC3	EDPC4	EDPC5	EDPC6	EDPC7	EDPC8	EDPC9	EDPC10	EDPC11	EDPC12	EDPC13	EDPC14	EDPC15
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	EDPC*x*	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.

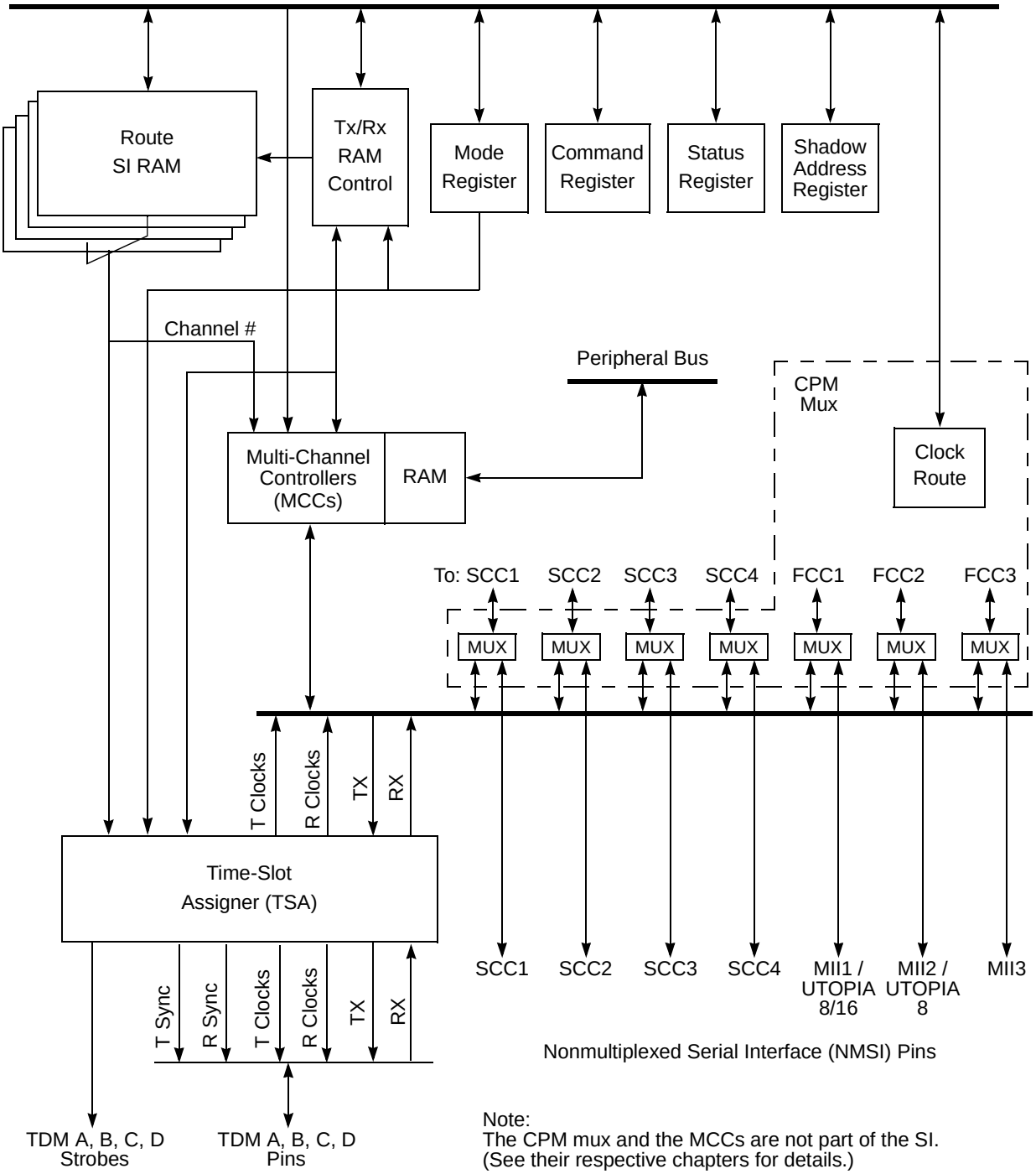


Figure 22-1. SI Block Diagram

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 serial-dependent:
  - For MCCs: $\frac{\text{CPM clock}}{7}$
  - For all other serials: $\frac{\text{CPM clock}}{3}$

NOTE

This clock ratio is based on the hardware architecture and does not ensure that an application will run at that speed. It is the responsibility of the system designer to check AC specifications of the I/O pins and determine the maximum frequency.

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 4 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.

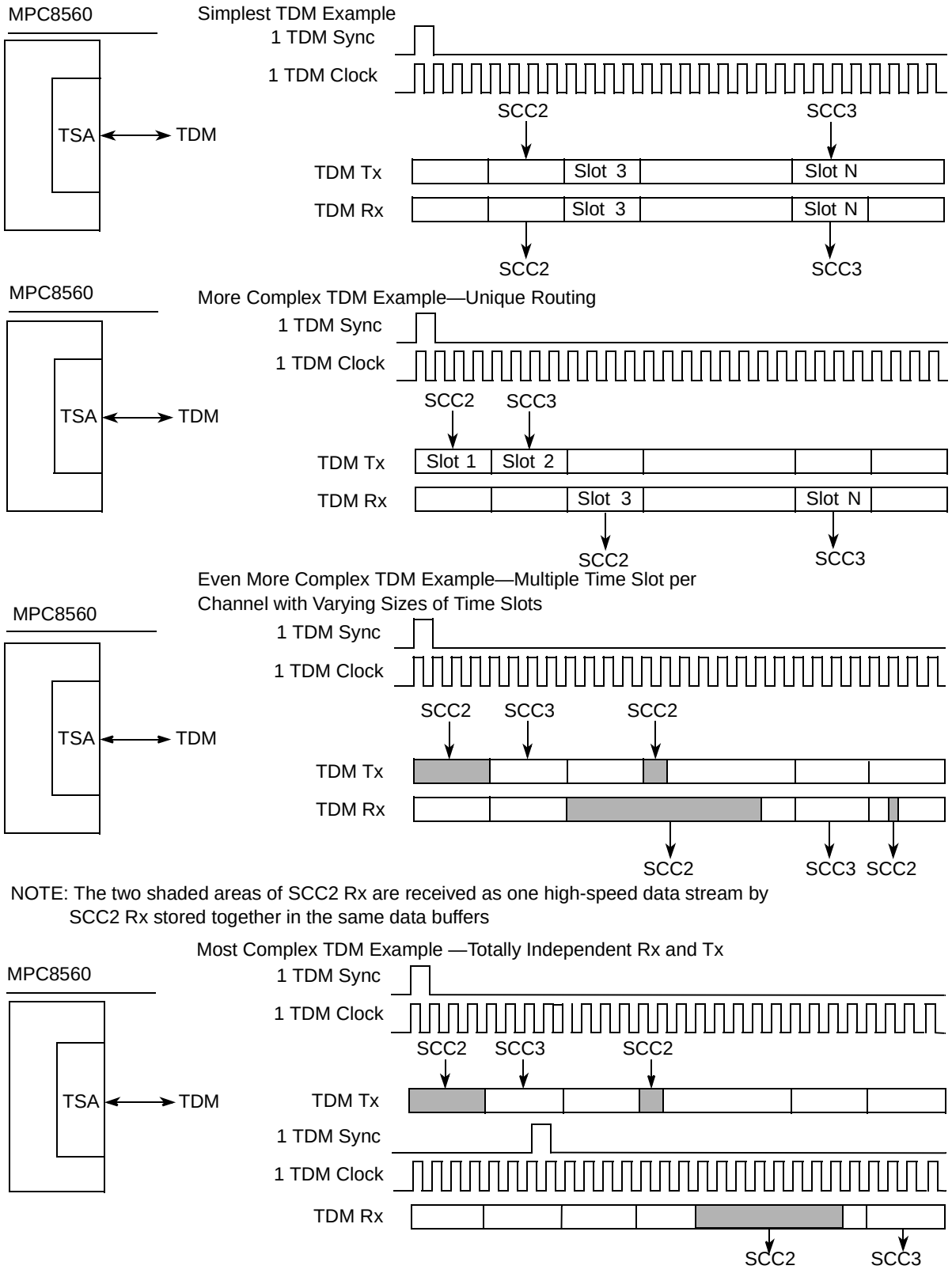
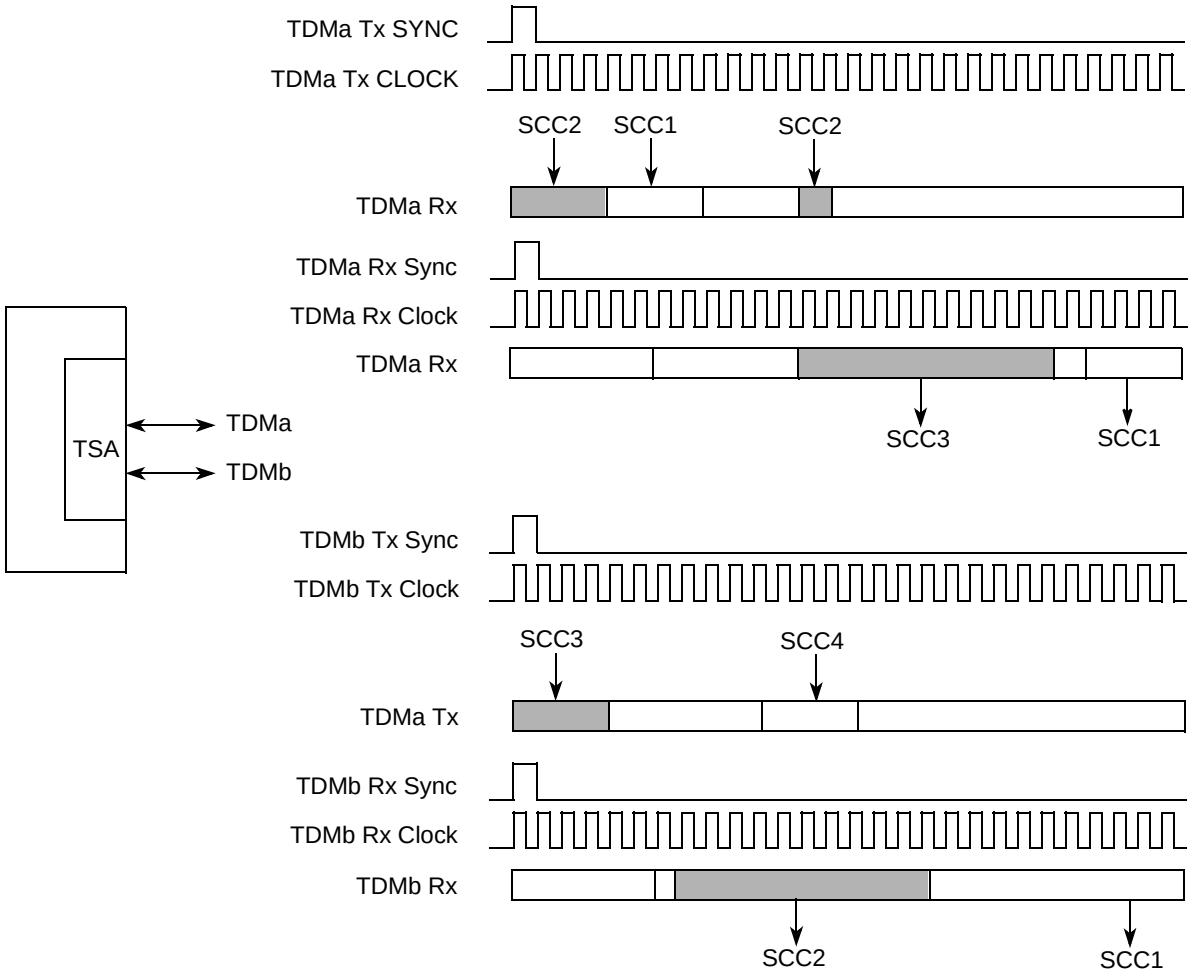


Figure 22-2. Various Configurations of a Single TDM Channel

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.

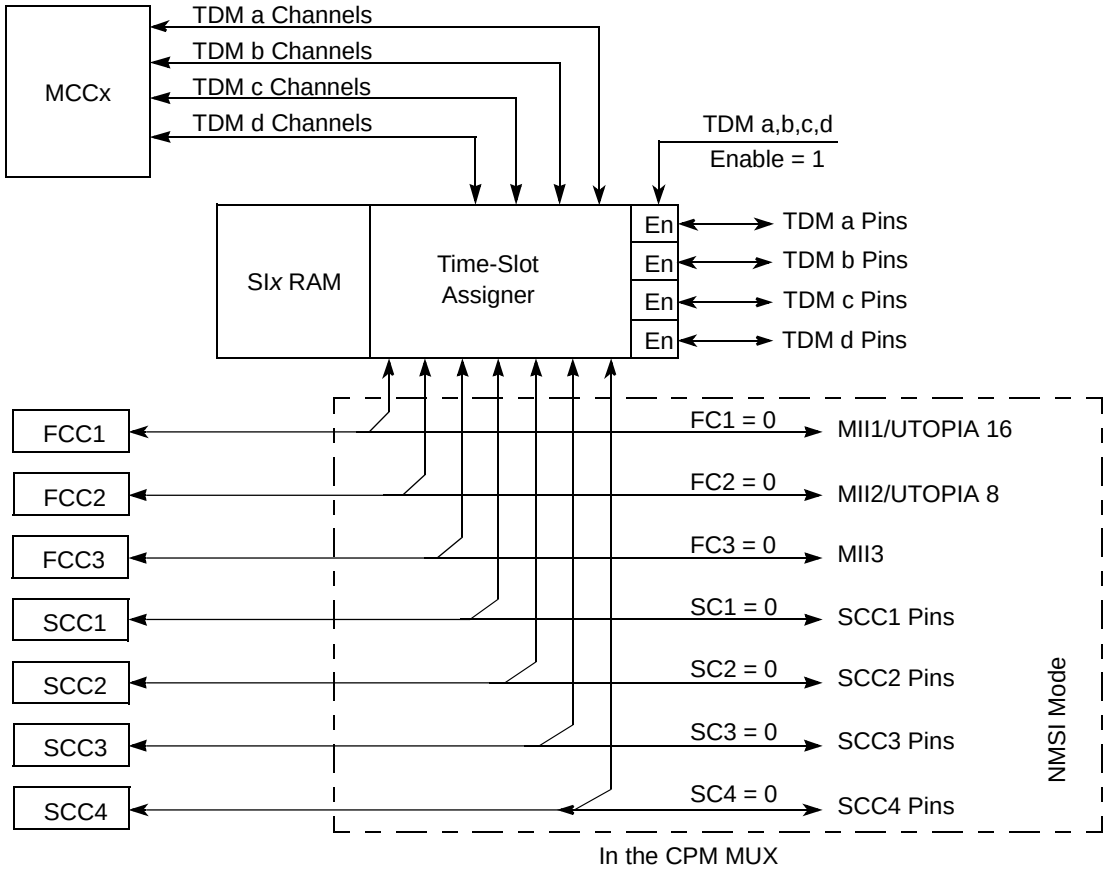


Figure 22-4. Enabling Connections to the TSA

22.4 Serial Interface RAM

Each SI has a transmit RAM and a receive RAM, each with 4 banks of 64 half-word 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.

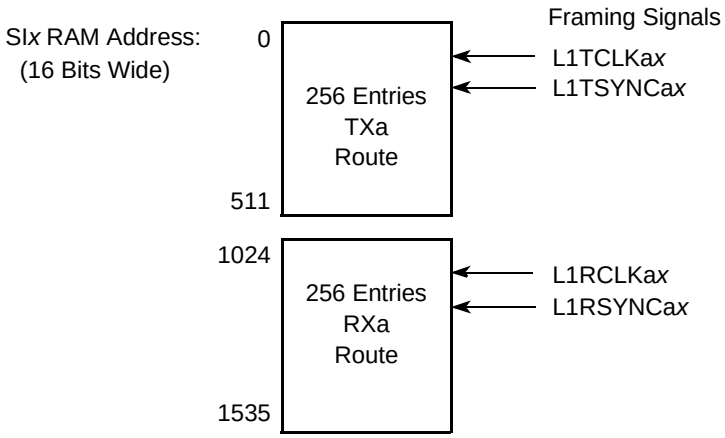


Figure 22-5. One TDM Channel with Static Frames and Independent Rx and Tx Routes

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.”

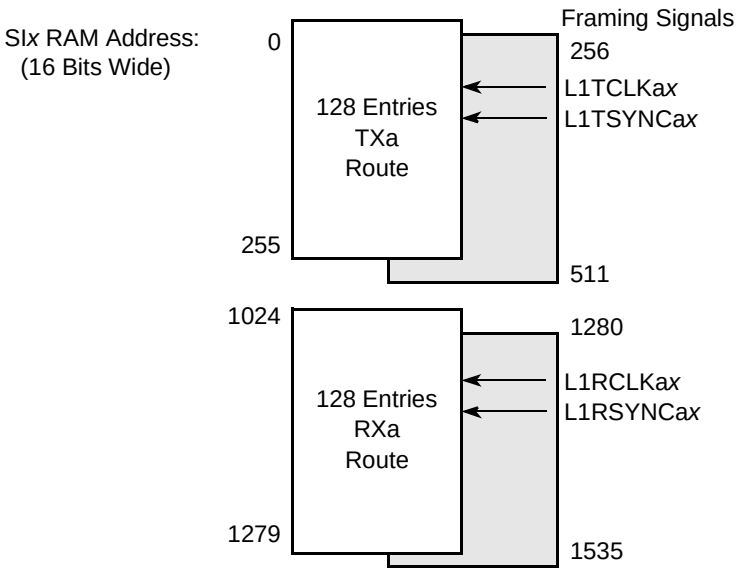


Figure 22-6. One TDM Channel with Shadow RAM for Dynamic Route Change

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.

The use of MCC slots is restricted for slots with lengths up to a single byte. For channels, that require more than a single byte, superchannel or slot splitting is mandatory.

	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 2, “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 and SCCs, 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	SSEL*x*	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 (L1RCLK*x*). 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. 0000 The 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

Table 22-1. SI*x* RAM Entry (MCC = 0) (continued)

Bits	Name	Description
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.
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 (that is, 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.

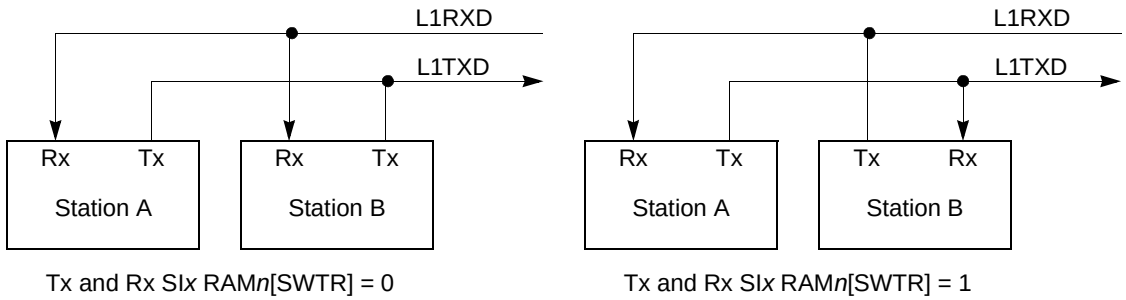


Figure 22-8. Using the SWTR Feature

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.

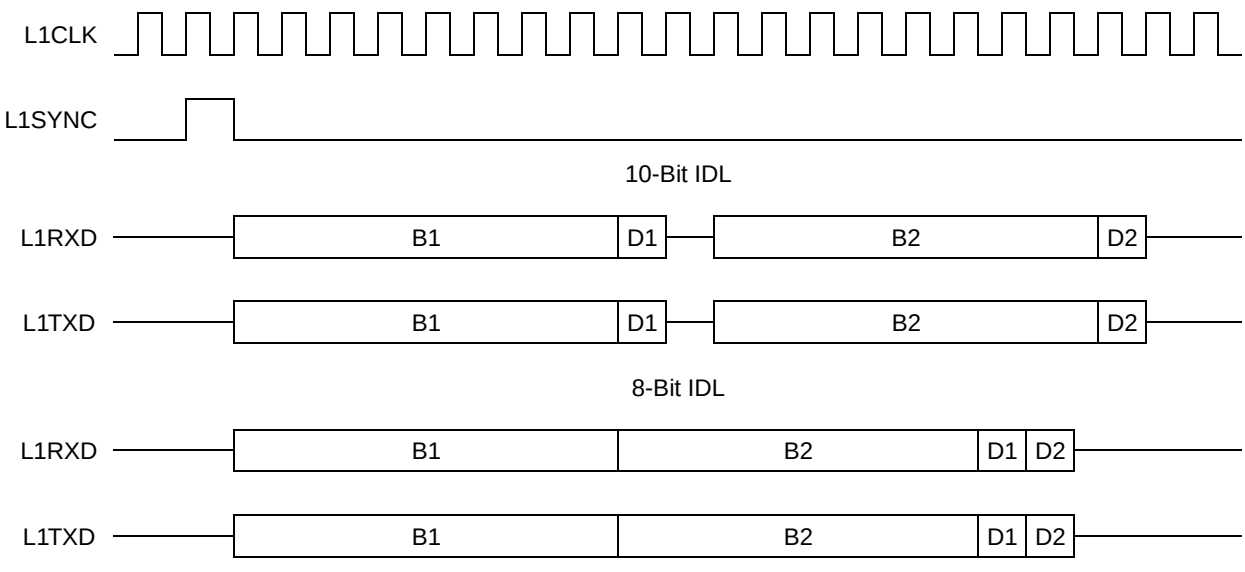
When MCC = 1, the SI*x* RAM entry fields function as described in Table 22-2.

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 (that is, 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
2	0	0	0000	0000	000	0	0	1-bit no support
3	0	0	0100	0000	011	0	0	4-bit strobe 2
4	0	0	0000	0011	011	0	0	4-bit SCC3
5	0	0	1000	0001	000	0	1	1-bit SCC1 strobe 1

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.

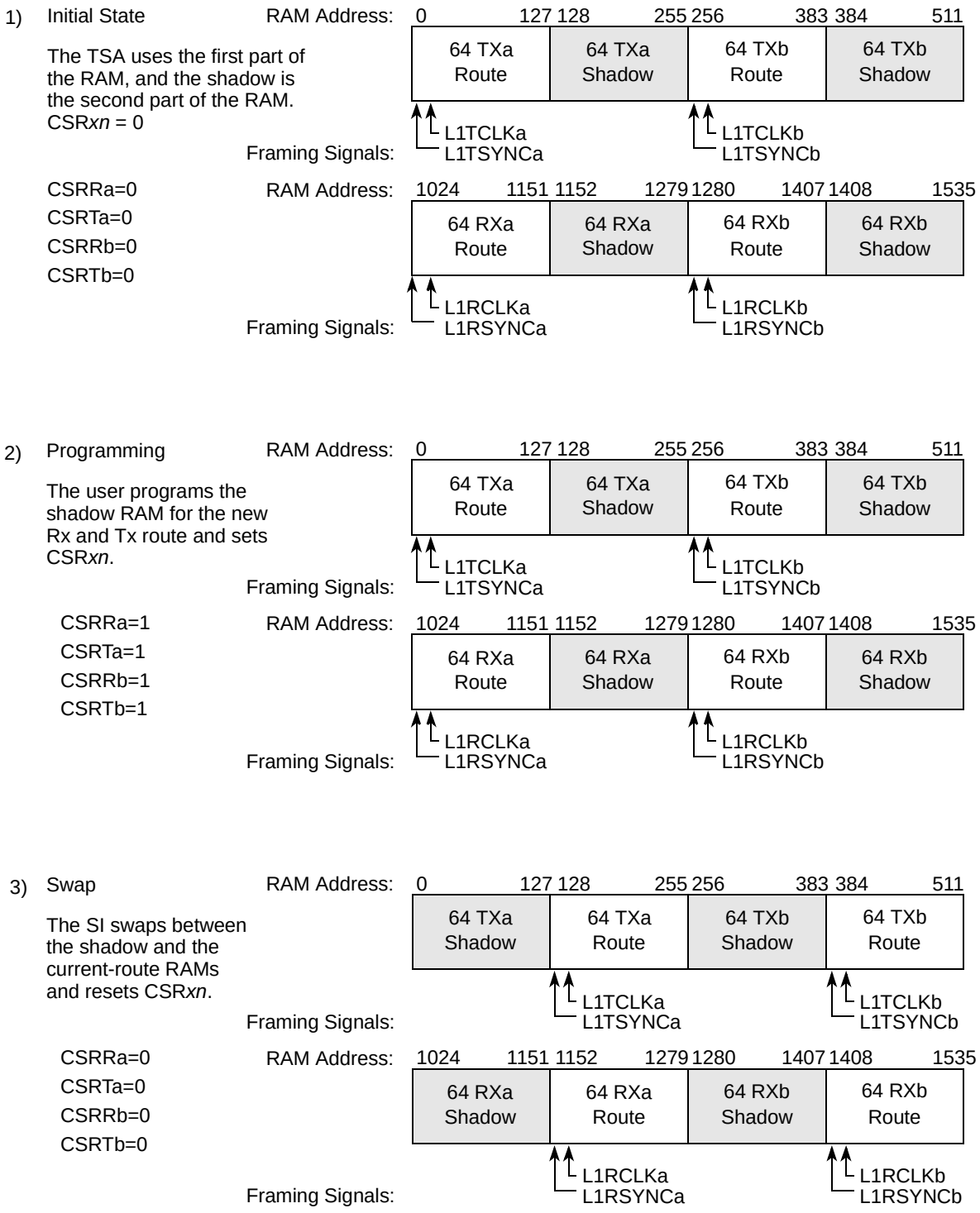


Figure 22-10. Example: SI*x* RAM Dynamic Changes, TDMa and b, Same SI*x* RAM Size

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

Bits	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. SI*x*MR 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.

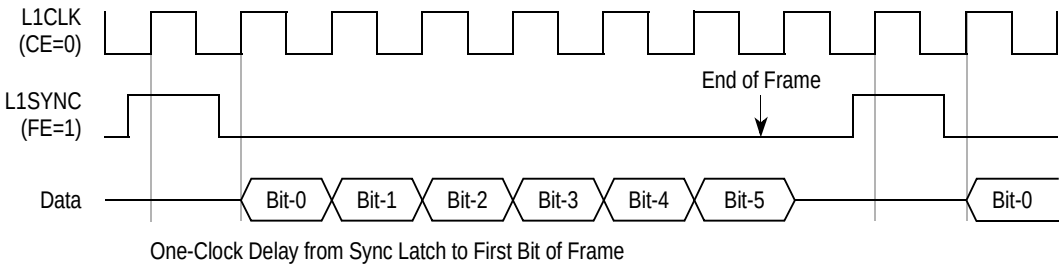


Figure 22-13. One-Clock Delay from Sync to Data (*x*FSD = 01)

Figure 22-14 shows the elimination of the single-clock delay shown in Figure 22-13 by clearing *x*FSD.

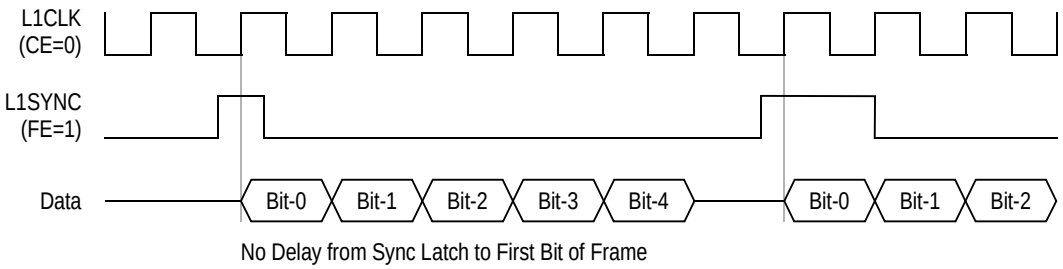


Figure 22-14. No Delay from Sync to Data (*x*FSD = 00)

Figure 22-15 shows the effects of changing FE when CE = 1 with a 1-bit frame sync delay.

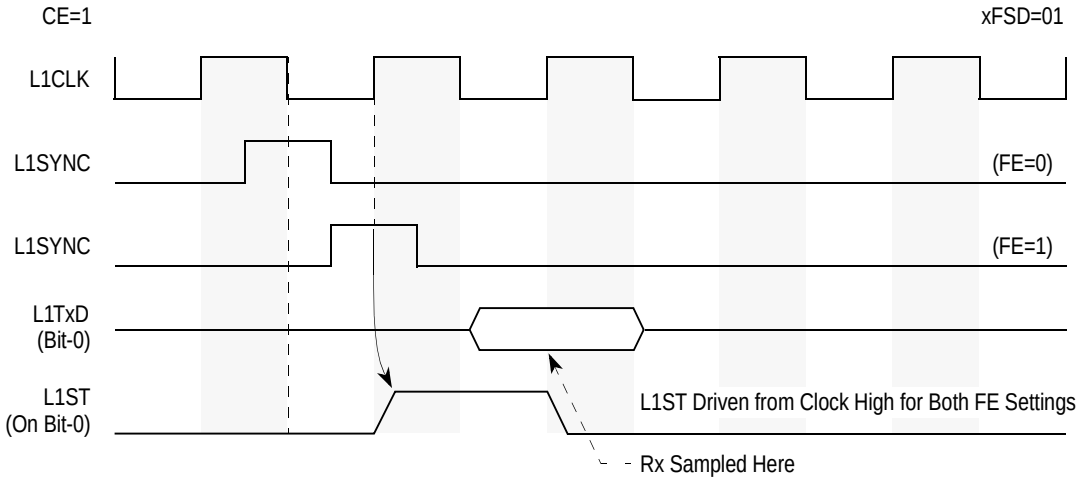


Figure 22-15. Falling Edge (FE) Effect When CE = 1 and *x*FSD = 01

Figure 22-16 shows the effects of changing FE when CE = 0 with a 1-bit frame sync delay.

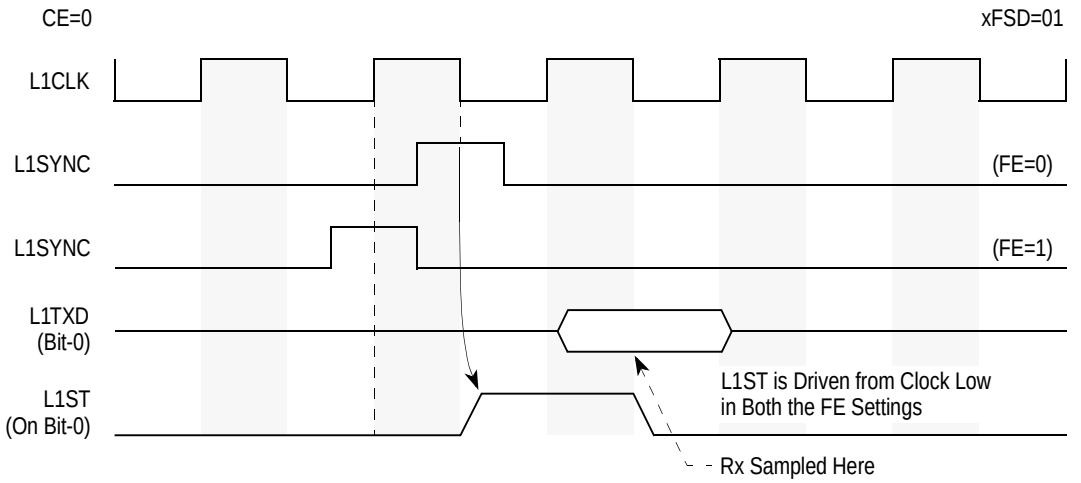


Figure 22-16. Falling Edge (FE) Effect When CE = 0 and *x*FSD = 01

Figure 22-17 shows the effects of changing FE when CE = 1 with no frame sync delay.

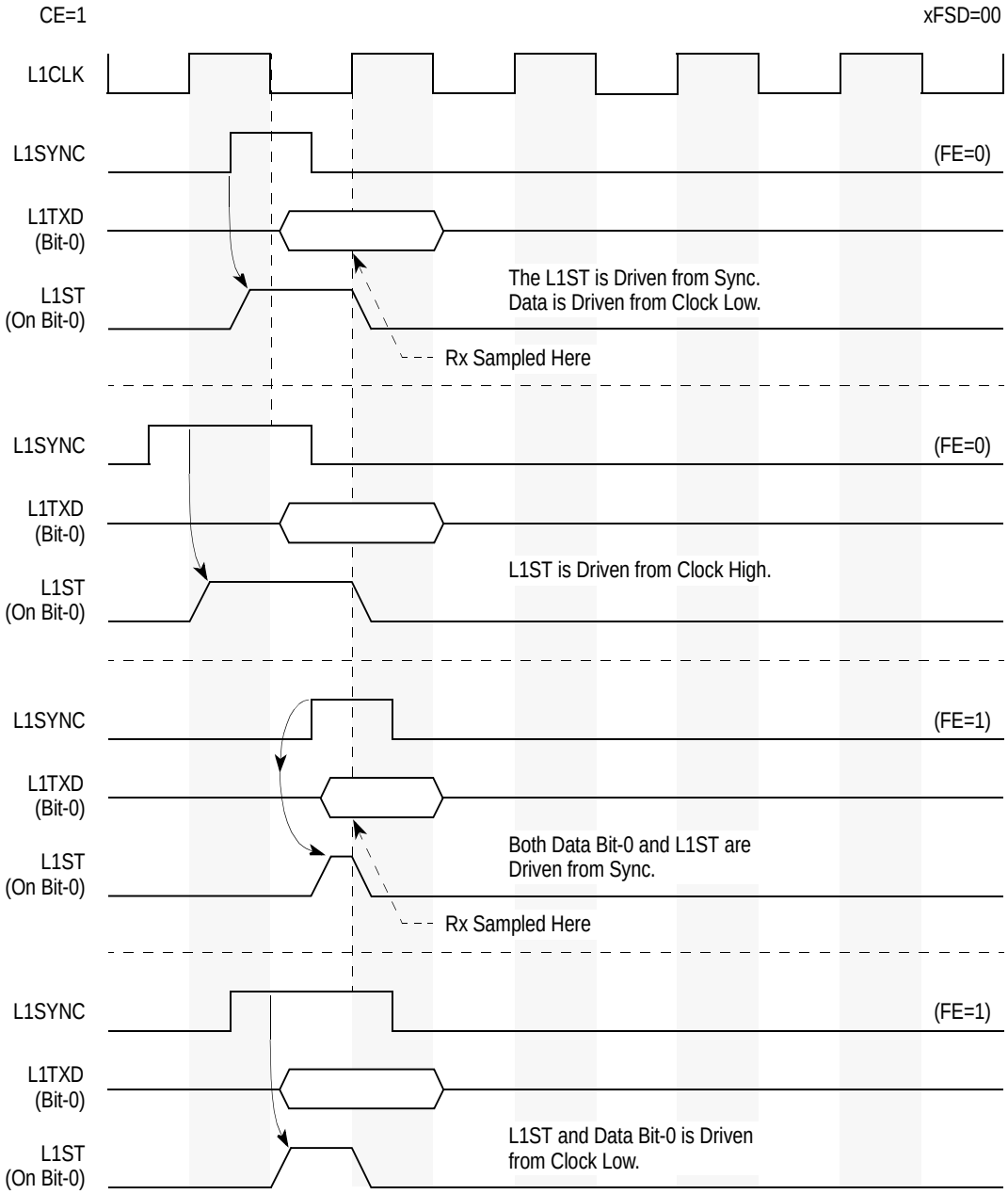


Figure 22-17. Falling Edge (FE) Effect When CE = 1 and *x*FSD = 00

Figure 22-18 shows the effects of changing FE when CE = 0 with no frame sync delay.

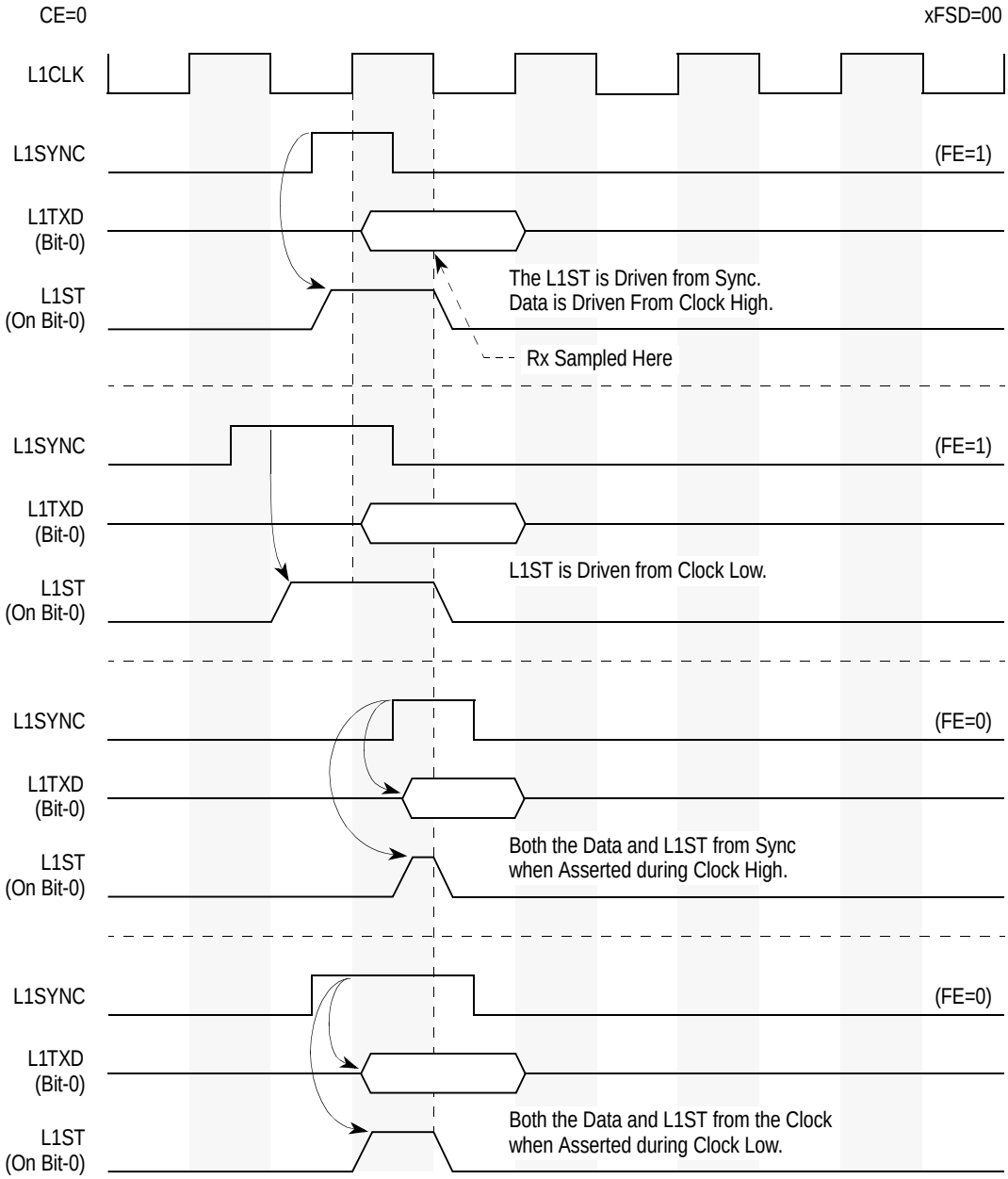
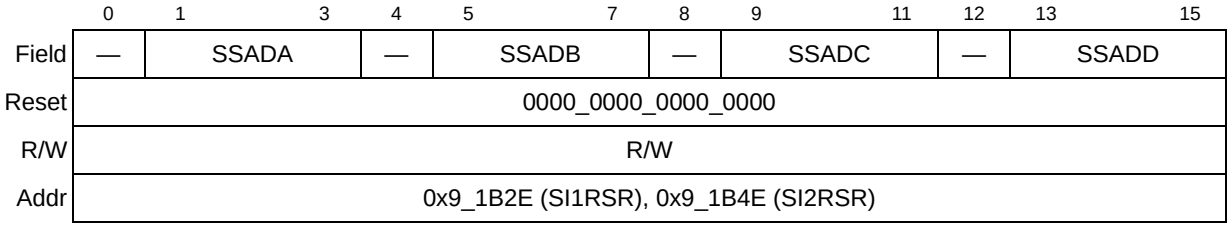


Figure 22-18. Falling Edge (FE) Effect When CE = 0 and *x*FSD = 00

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)

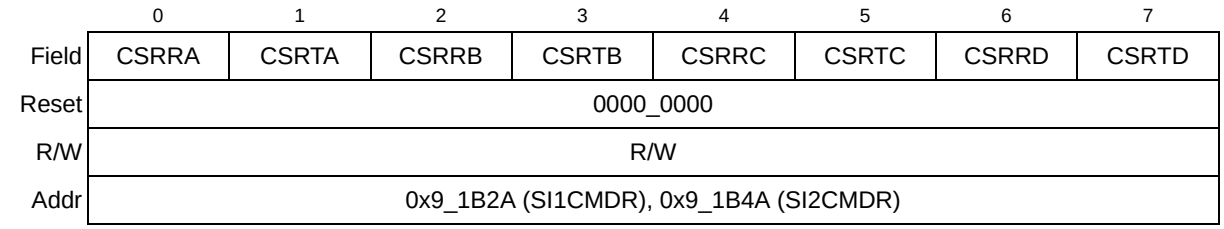
Table 22-6 describes SI*x*RSR fields.

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. Setting 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. Setting 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	CROR*x*	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	CROT*x*	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.

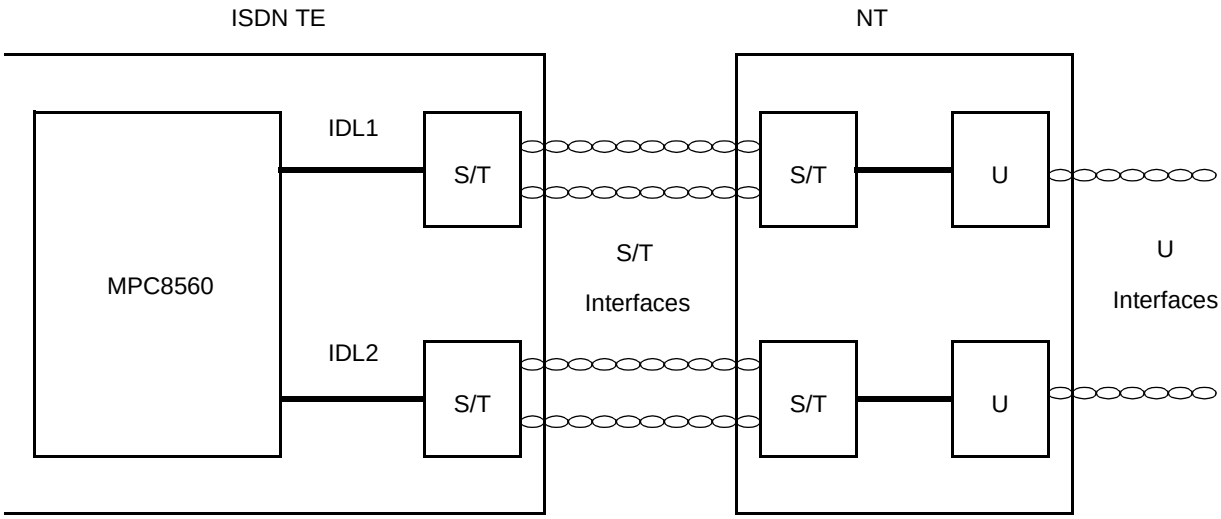


Figure 22-22. Dual IDL Bus Application Example

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 45.2.1, “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	SIx 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 TDMa (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.

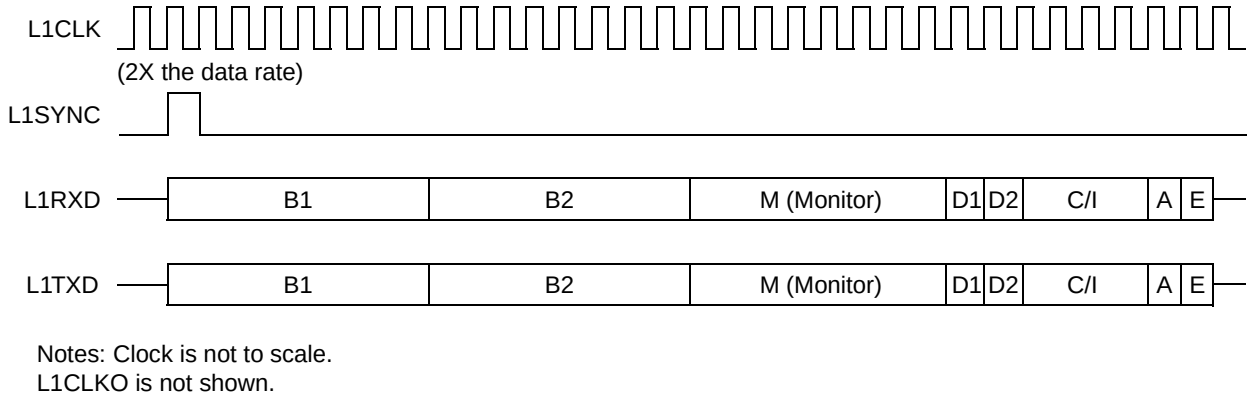
In the GCI bus, the clock rate is twice the data rate. The SI divides the input clock by two to produce the data clock. The MPC8560 also has data strobe lines and the 1× data rate clock L1CLKOx output pins. These signals are used for interfacing devices to GCI that do not support the GCI bus. Table 22-10 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.

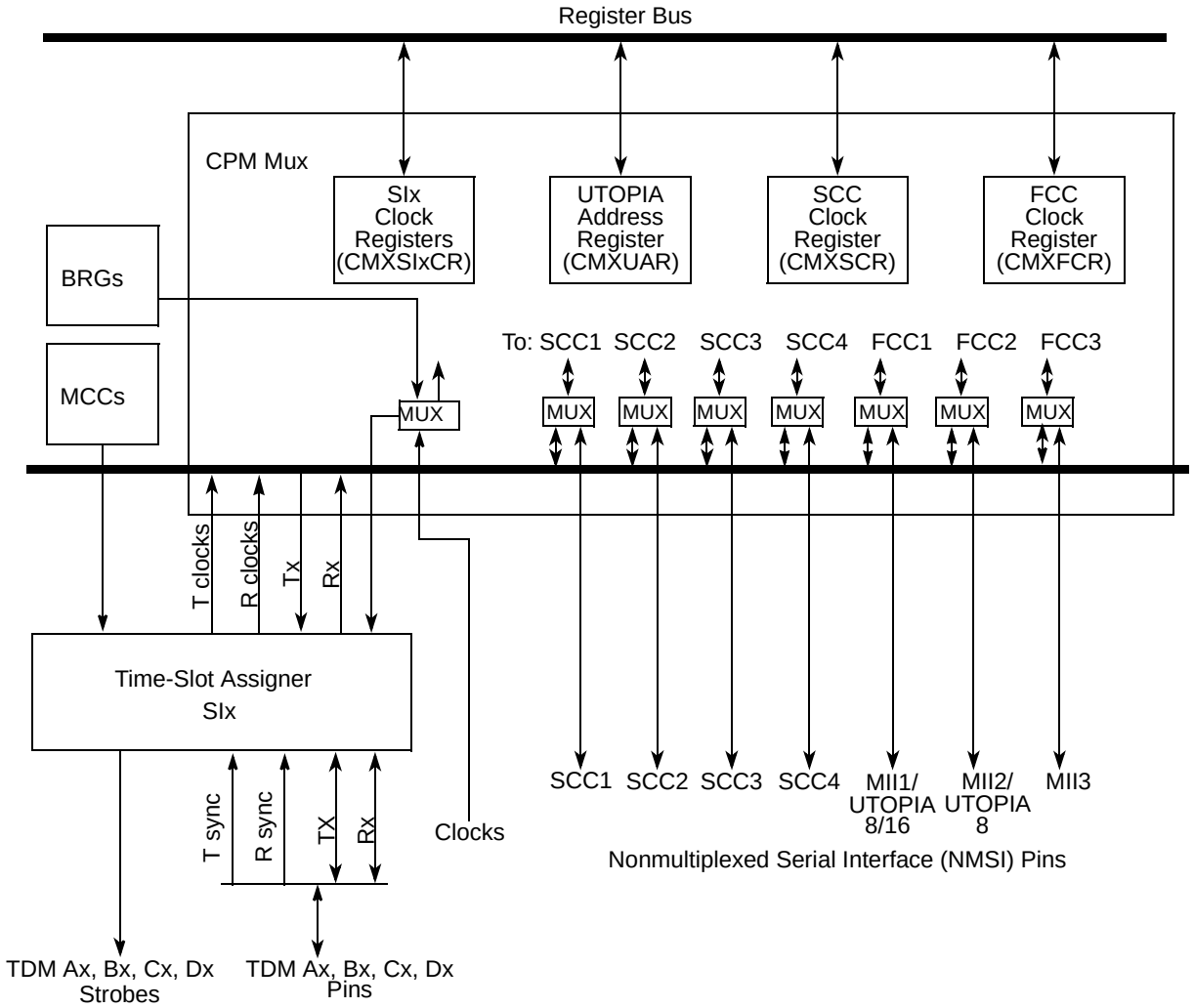


Figure 23-1. CPM Multiplexing Logic (CMX) Block Diagram

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 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.

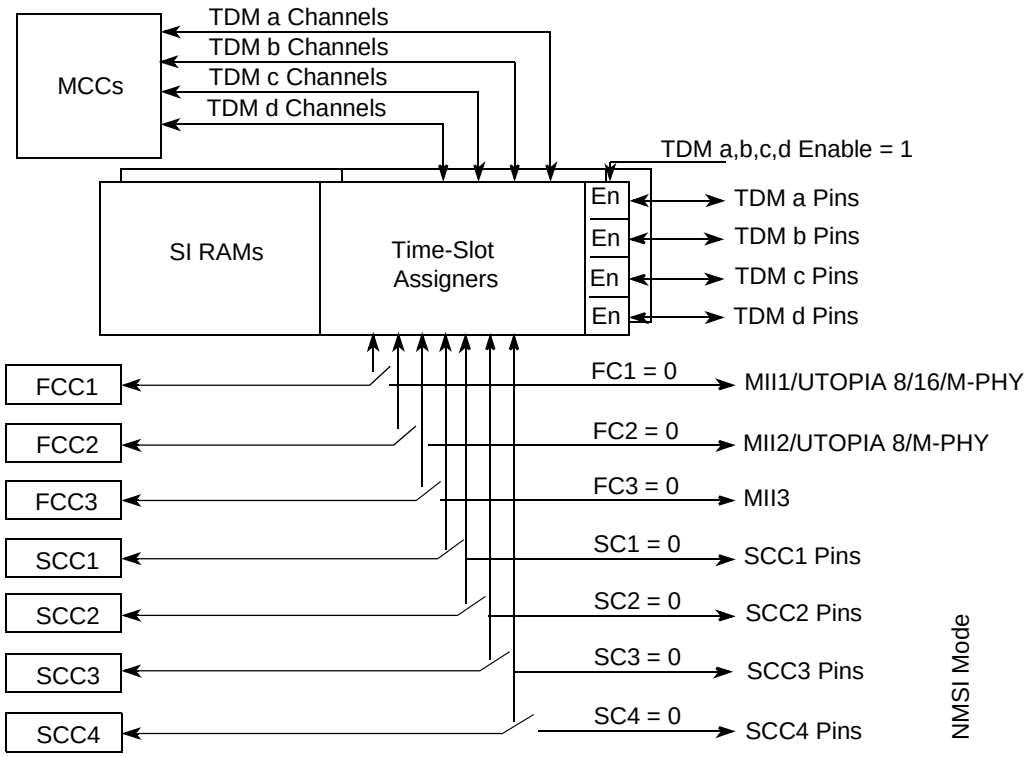


Figure 23-2. Enabling Connections to the TSA

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.

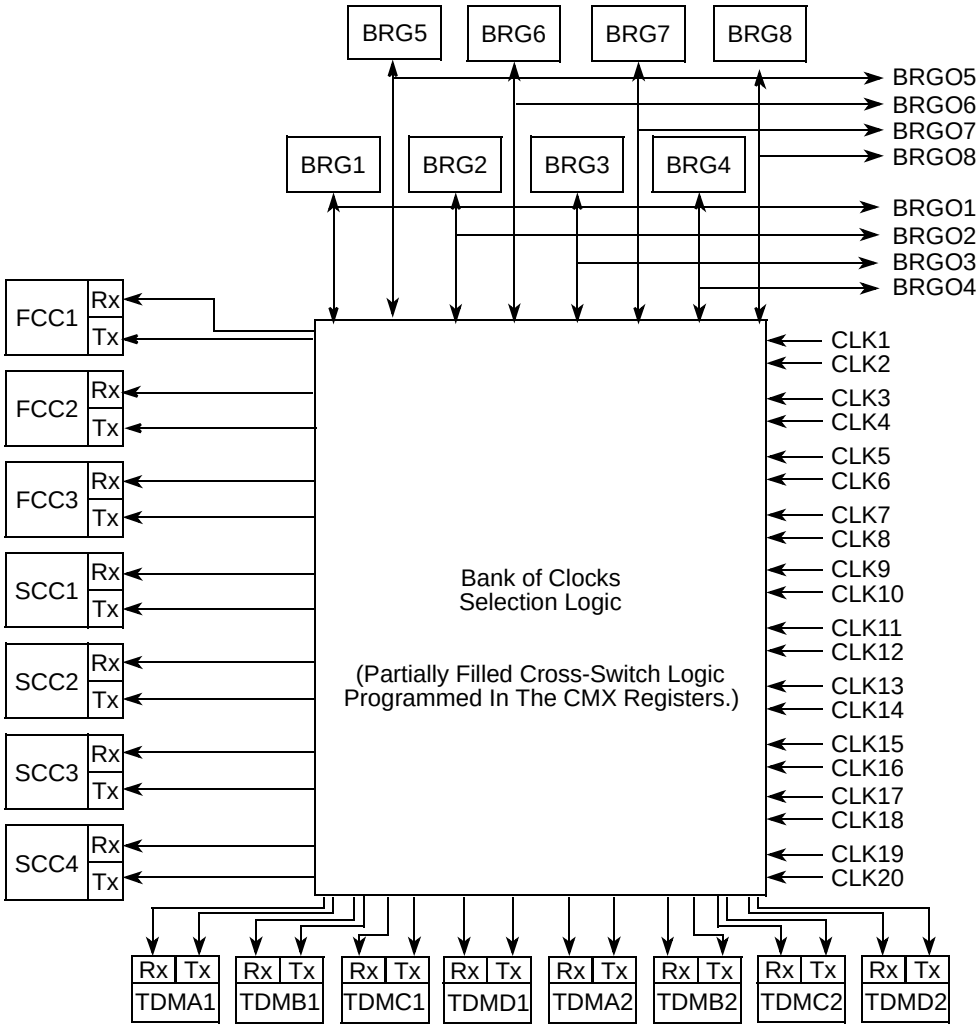


Figure 23-3. Bank of Clocks

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 45, "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	MAD4–MAD3	Master address output pin *x* connection. Note that the address indexes are relative to FCC1; see Figure 23-7. These bits determine the number of ATM PHYs supported by FCC1 and FCC2. 00 FCC1 supports 7 PHYs; FCC2 supports 31 PHYs 01 FCC1 supports 15 PHYs; FCC2 supports 15 PHYs 10 Reserved 11 FCC1 supports 31 PHYs; FCC2 supports 7 PHYs

Table 23-2. CMXUAR Field Descriptions (continued)

Bits	Name	Description
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.
10–11	F2IRB	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.

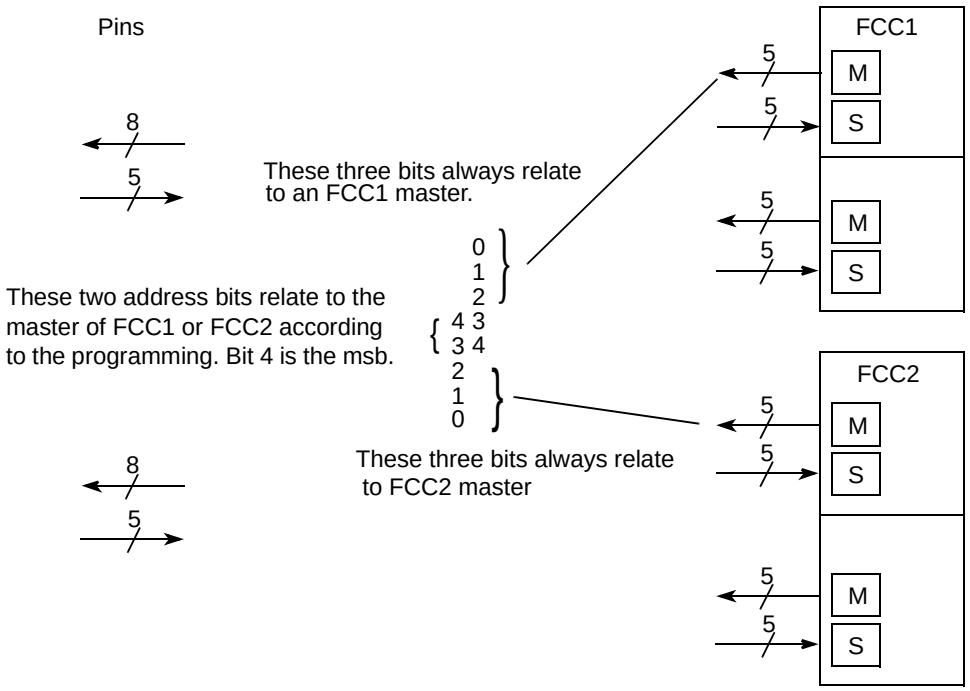
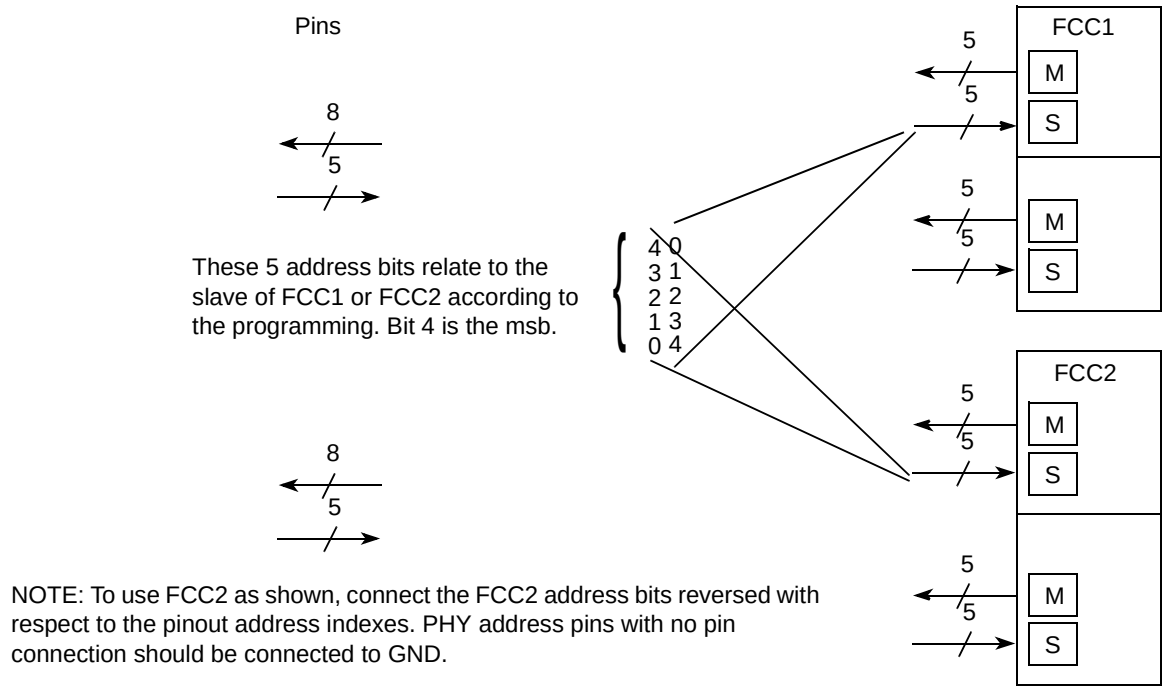


Figure 23-5. Connection of the Master Address

- 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.



NOTE: To use FCC2 as shown, connect the FCC2 address bits reversed with respect to the pinout address indexes. PHY address pins with no pin connection should be connected to GND.

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.

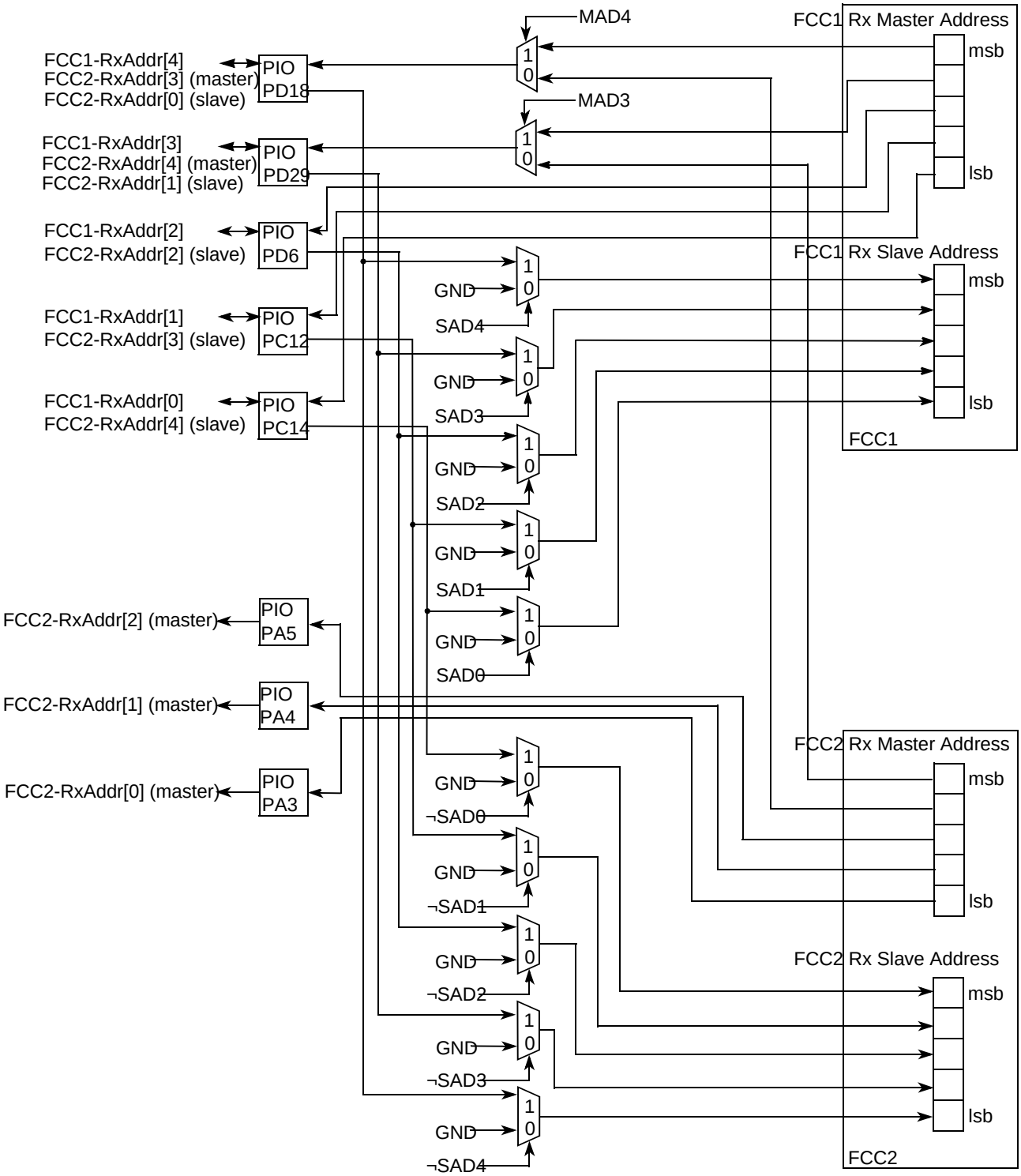


Figure 23-7. Multi-PHY Receive Address Multiplexing

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.

Table 23-5. CMXFCR Field Descriptions (continued)

Bits	Name	Description
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.
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.

Table 23-5. CMXFCR Field Descriptions (continued)

Bits	Name	Description
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.
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.

Table 23-6. CMXSCR Field Descriptions (continued)

Bits	Name	Description
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.
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.

Table 23-6. CMXSCR Field Descriptions (continued)

Bits	Name	Description
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.

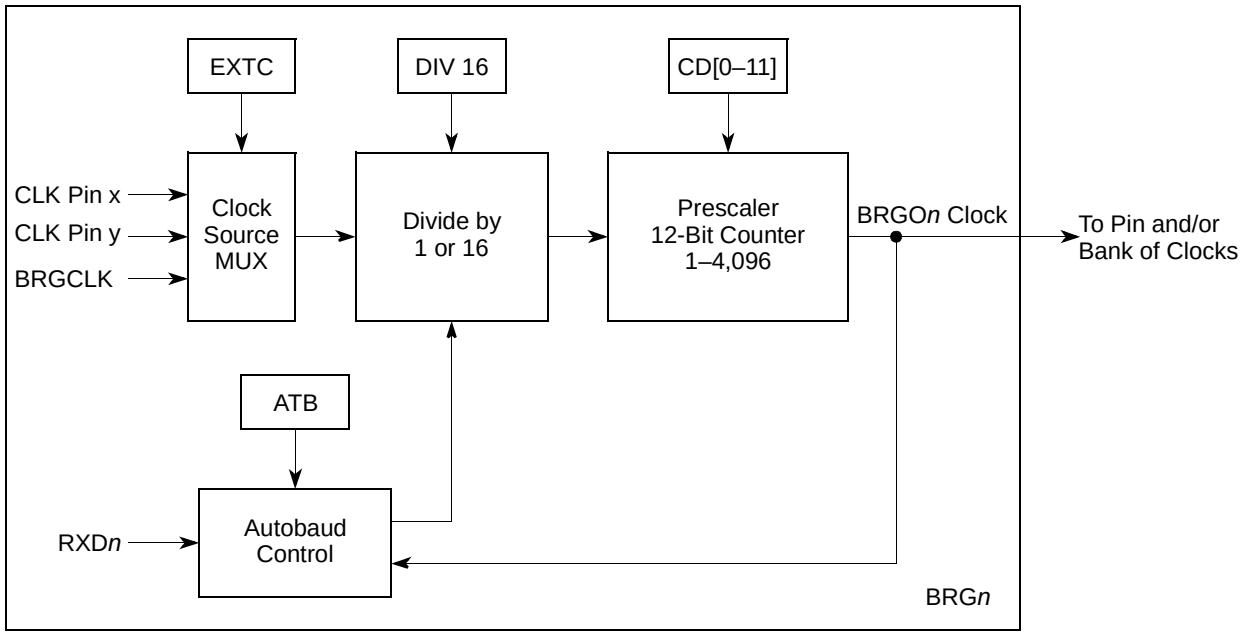


Figure 24-1. Baud-Rate Generator (BRG) Block Diagram

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 internal space. The SCCR defines the BRGCLK frequency to be supplied to the BRGs.

	0	15
Field	—	
Reset	—	
R/W	R/W	
Addr	0x9_0C80	

	16	29	30	31
Field	—		DFBRG	
Reset	—		01	
R/W	R/W			
Addr	0x9_0C82			

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 24.1, “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 66 MHz (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:

$$\text{Async Baud Rate} = \frac{\text{BRGCLK or External Clock Source}}{(\text{Prescale Divider}) \times (\text{Clock Divider} + 1) \times (\text{Sampling Rate})}$$

$$= \frac{\text{BRGCx[EXTC]}}{(\text{BRGCx[DIV16]}) \times (\text{BRGCx[CD]} + 1) \times (\text{GSMRx\_L[xDCR]})}$$

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 formula:

$$\text{Sync Baud Rate} = \frac{\text{BRGCLK or External Clock Source}}{(\text{Prescale Divider}) \times (\text{Clock Divider} + 1)}$$

$$= \frac{\text{BRGCx[EXTC]}}{(\text{BRGCx[DIV16]}) \times (\text{BRGCx[CD]} + 1)}$$

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.

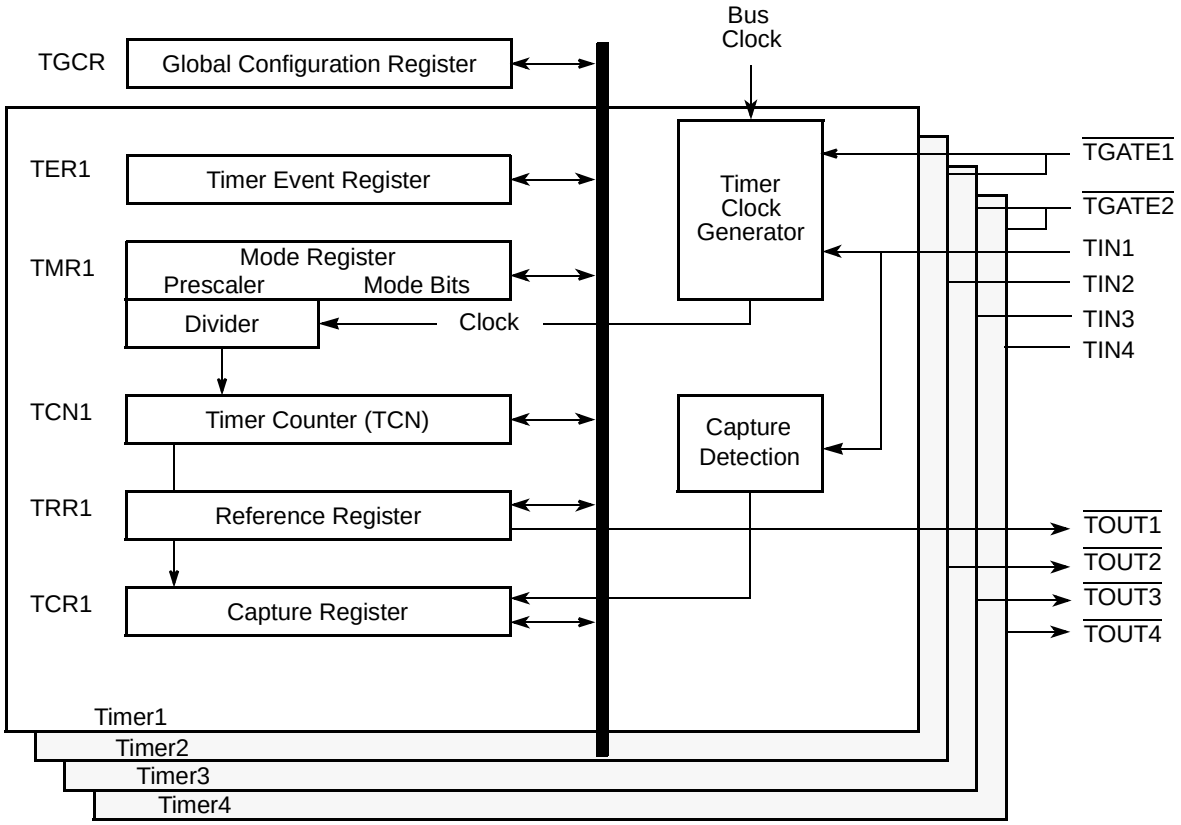


Figure 25-1. Timer Block Diagram

Pin assignments for TIN*x*, $\overline{\text{TGATE}}$*x*, and $\overline{\text{TOUT}}$*x* are described in Section 45.5, “Ports Tables.”

25.1 Features

The key features of the timer include the following:

- The maximum input clock is the internal CPM clock which is the core complex bus (CCB) clock divided by 3.
- Maximum period of 2.6 seconds (at 100 MHz)
- 10-ns resolution (at 100 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:

- Internal CPM clock (CCB/3)
- Internal CPM clock divided by 16 (CCB/48)
- Corresponding TIN*x*, programmed in the parallel port registers

The internal CPM clock is generated in the CPM clock synthesizer and defaults to the CCB clock frequency divided by 3. 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 (10 ns at 100 MHz). The maximum period (when the reference value is all ones) is 268,435,456 cycles (2.6 seconds at 100 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 internal CPM clock. After the falling edge of $\overline{\text{TGATE}x}$ is recognized, the counter begins counting after one internal CPM clock cycle 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.

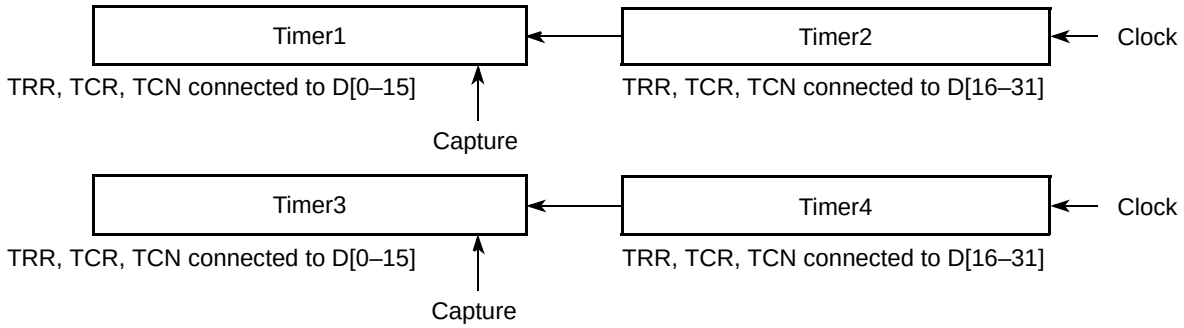


Figure 25-2. Timer Cascaded Mode Block Diagram

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[1]	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.
2	**STP2**	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.

Table 25-1. TGCR1 Field Descriptions (continued)

Bits	Name[1]	Description
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.

1 **Boldfaced** entries must be initialized by the user.

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[1]	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.
2	**STP4**	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.

Table 25-2. TGCR2 Field Descriptions (continued)

Bit	Name[1]	Description
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.

1 **Boldfaced** entries must be initialized by the user.

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[1]	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.

1 **Boldfaced** entries must be initialized by the user.

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.

	0 … 15
Field	**Timeout Reference Value**
Reset	0xFFFF
R/W	R/W
Addr	0x9_0D94 (TRR1), 0x9_0D96 (TRR2), 0x9_0DA4 (TRR3), 0x9_0DA6 (TRR4)

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.

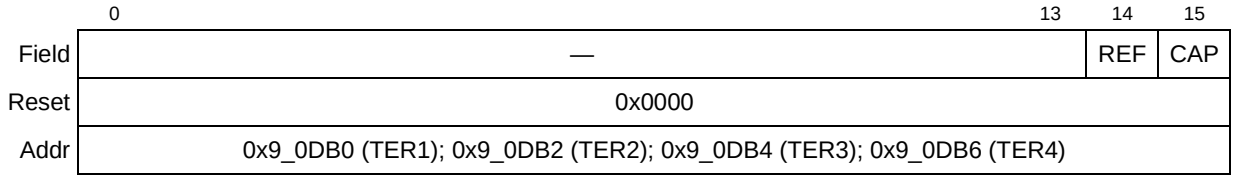


Figure 25-9. Timer Event Registers (TER1–TER4)

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).

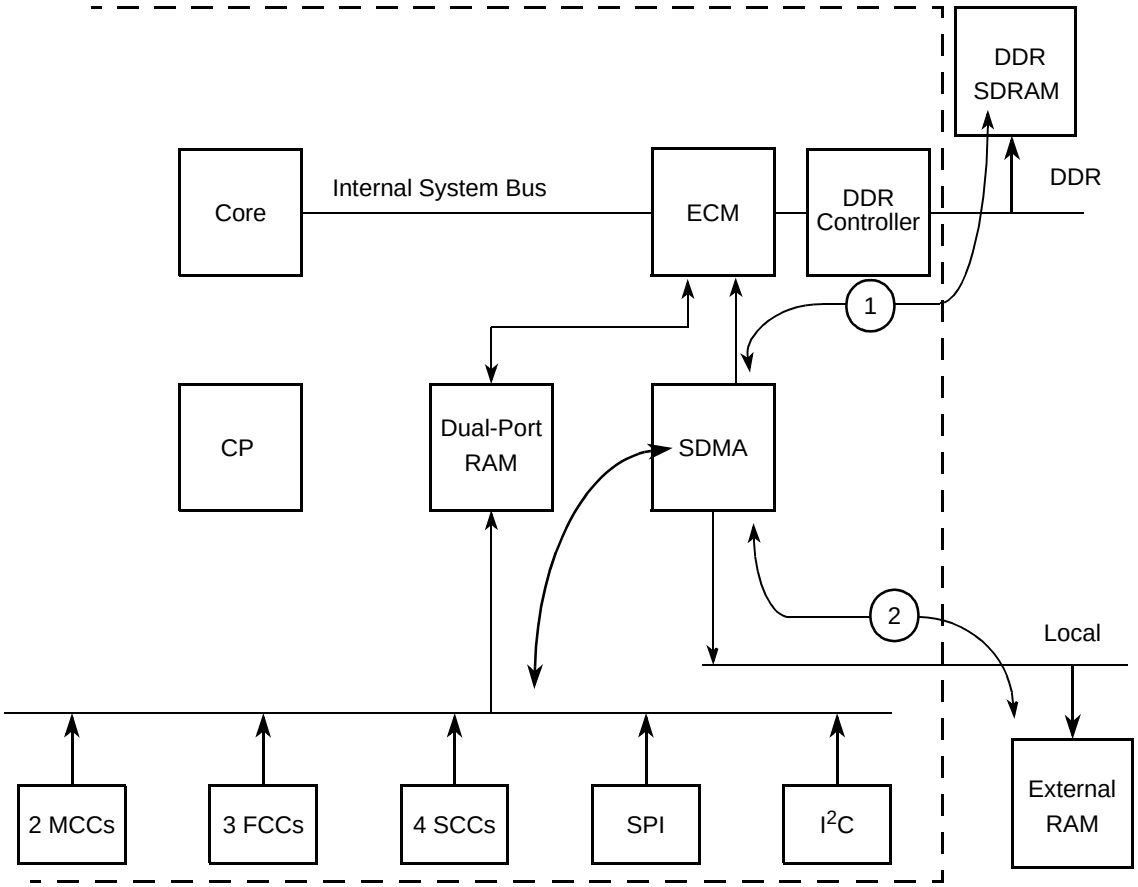


Figure 26-1. SDMA Data Paths

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 always operates in big-endian format when accesses data.

If a system or local bus error occurs on a CP-related access by the SDMA, the CP generates an interrupt which will set the SDMA SYS or SDMA LCL bit in the SIPNR_L register. Section 21.5.1.3, “CPM Interrupt Pending Registers (SIPNR_H and SIPNR_L).” The interrupt service routine then reads the appropriate DMA transfer error address register (SMAER for the system bus or LMAER 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 SMEVR or LMEVR.

26.1 SDMA Registers

26.1.1 SDMA Address Error Registers (SMAER and LMAER)

These registers indicate the address of the error that occurred at a read or write transaction initiated by the system or local channel SDMA. The SMAER holds the system address of the system SDMA-initiated error transaction, and the LMAER holds the local address of the local SDMA-initiated error transaction. SMAER and LMAER are cleared by hard reset. They are not affected by soft reset. These registers are updated at the first occurrence of the error detection; the next capture is enabled only after the event register is cleared.

	0	31
Field	SMAER/LMAER	
R/W	R	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
Addr	0x9_0050 (SMAER); 0x9_0060 (LMAER)	

Figure 26-2. SDMA Address Error Registers (SMAER and LMAER)

26.1.2 SDMA Event Registers (SMEVR and LMEVR)

These registers indicate if the error information that is found in SMAER/LMAER is valid. They are updated in the first SDMA data error that is detected. Bit 0 is cleared by writing a 1 to it. Writing a 0 has no effect. MSNUM contains the serial number of the channel that created the data error. The bit fields of these two registers are updated only when a read error occurs. SMEVR and LMEVR are cleared by hard reset. Soft reset has no effect.

	0	1	2–7	8–31
Field	DATAERR	—	MSNUM	—
R/W	R/W		R	R/W
Reset	0000_0000_0000_0000_0000_0000_0000_0000			
Addr	0x9_0058 (SMEVR); 0x9_0068 (LMEVR)			

Figure 26-3. SDMA Event Registers (SMEVR and LMEVR)

Table 26-1. SMEVR and LMEVR Field Descriptions

Bits	Name	Description
0	DATAERR	Indicates Data error, Error details are captured in SMAER/LMAER registers, and in MSNUM field. Bit is cleared by writing a 1 to it. Writing a 0 has no effect. 0 No error occurred 1 Error occurred while reading/writing data
1	—	Reserved
2–7	MSNUM	Represent the serial number of the channel that created the data error.
8–31	—	Reserved

26.1.3 SDMA Control Registers (SMCTR and LMCTR)

The SDMA Control Registers (SMCTR and LMCTR), shown in Figure 26-4, contain control parameters and can be updated through SREG.

	0	1	2–4	5–31
Field	ERR_MASK	—	—	—
R/W	R/W			
Reset	0011_1000_0000_0000_0000_0000_0000_0000			
Addr	0x9_005C (SMCTR); 0x9_006C (LMCTR)			

Figure 26-4. SDMA Control Registers (SMCTR and LMCTR)

Table 26-2. SMCTR/LMCTR Field Descriptions

Bits	Name	Description
0	ERR_MASK	Data Error Interrupt mask 0 Disable Error Interrupt 1 Enable Error Interrupt
1	—	Reserved

Table 26-2. SMCTR/LMCTR Field Descriptions (continued)

Bits	Name	Description
2–4	—	Reserved. **Note:** User must not change default value of bits 2–4 (‘111’) when writing to SMCTR/LMCTR.
5–31	—	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.

NOTE

This clock ratio is based on the hardware architecture and does not ensure that an application will run at that speed. It is the responsibility of the system designer to check AC specifications of the I/O pins and determine the maximum frequency.

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.

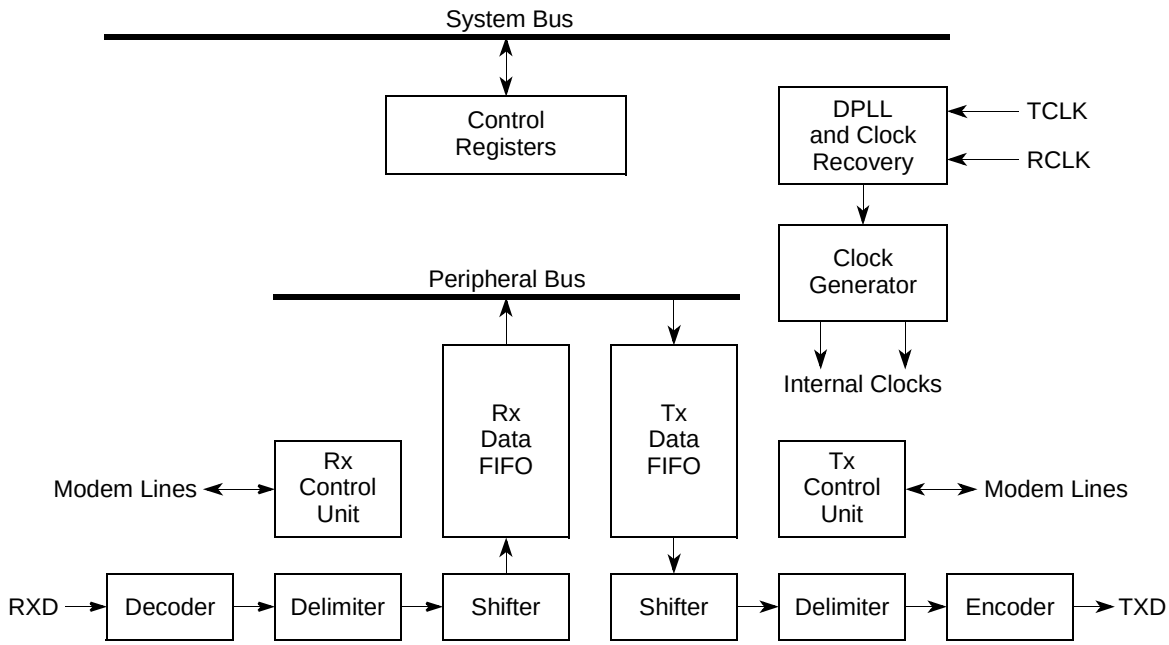


Figure 27-1. SCC Block Diagram

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 Freescale-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 … 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	9	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	19	20	21	22	23	24	25	26	27	28	29	30	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). 101 64 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 27.4.7, “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 27.4.7, "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 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.

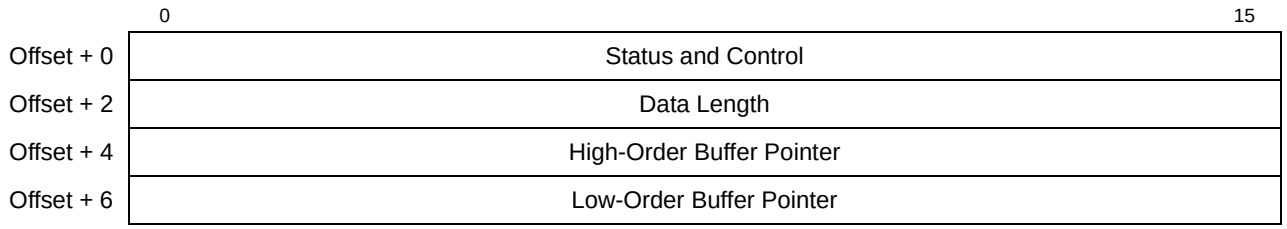


Figure 27-6. SCC Buffer Descriptors (BDs)

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.

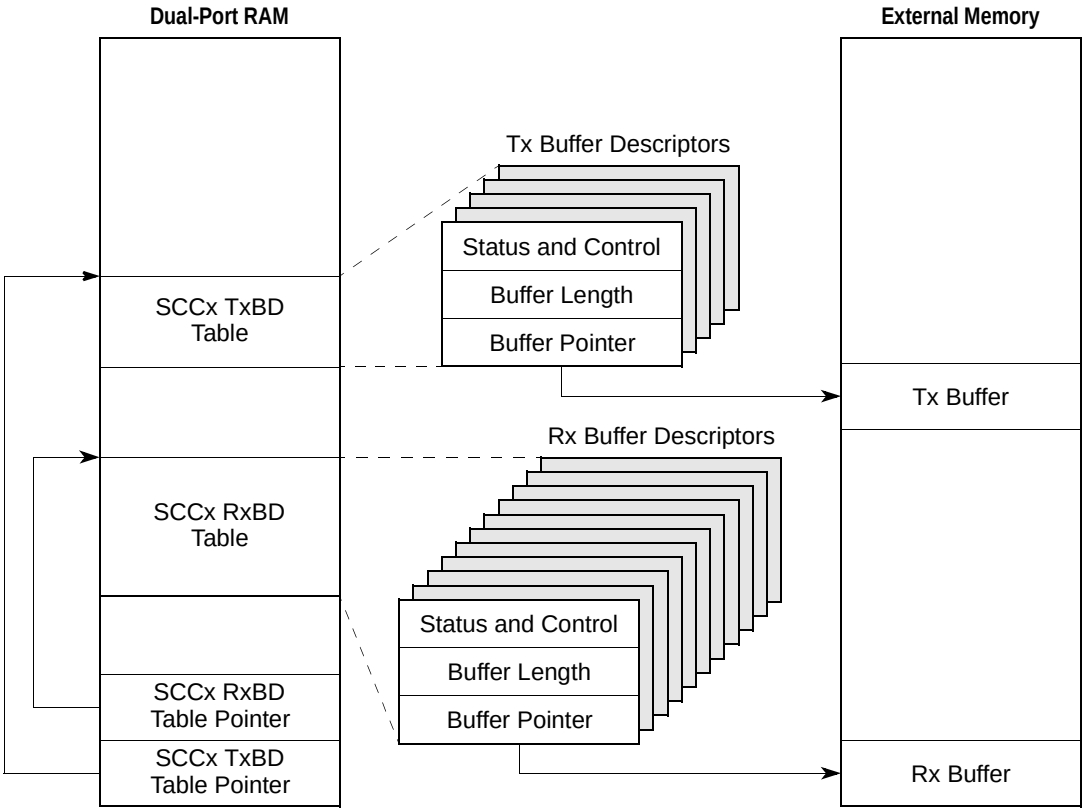


Figure 27-7. SCC BD and Buffer Memory Structure

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 27.4.7, "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[2]	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)

Offset [1]	Name[2]	Width	Description
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[4]. 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 [4]. 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 [4]. 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 [4]. 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 [4]
0x2C	TCRC	Word	Temp transmit CRC [4]
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 **Boldfaced** entries must be initialized by the user.
3 For CP use only
4 These parameters need not be accessed for normal operation but may be helpful for debugging.

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 I^2C. 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.

	0	1	2	3	4	5	6	7
Field	—		**GBL**	**BO**		—	**DTB**	—
Reset	0000_0000_0000_0000							
R/W	R/W							
Addr	SCC*x* base + 0x04 (RFCR*x*); SCC*x* base + 0x05 (TFCR*x*)							

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[1]	Description
0–1	—	Reserved, should be cleared.
2	**GBL**	Global 0 Snooping disabled. 1 Snooping enabled.

Table 27-6. RFCR*x* /TFCR*x* Field Descriptions (continued)

Bits	Name[1]	Description
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.

1 **Boldfaced** entries must be initialized by the user.

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 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 CPM 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].

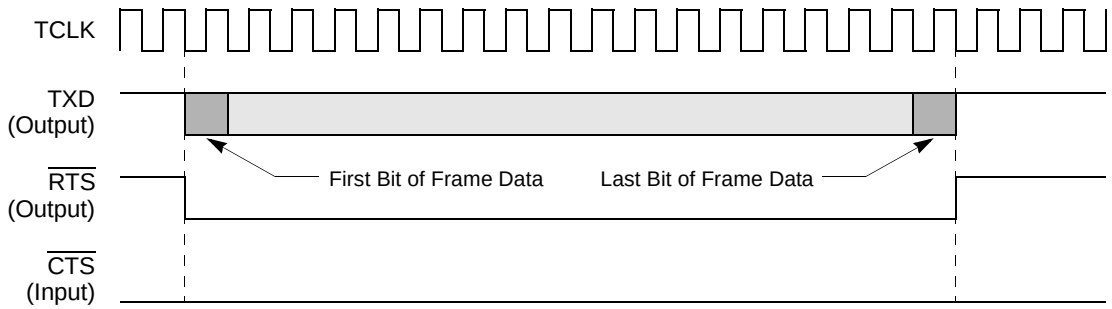
27.4.5 Controlling SCC Timing with $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$

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].

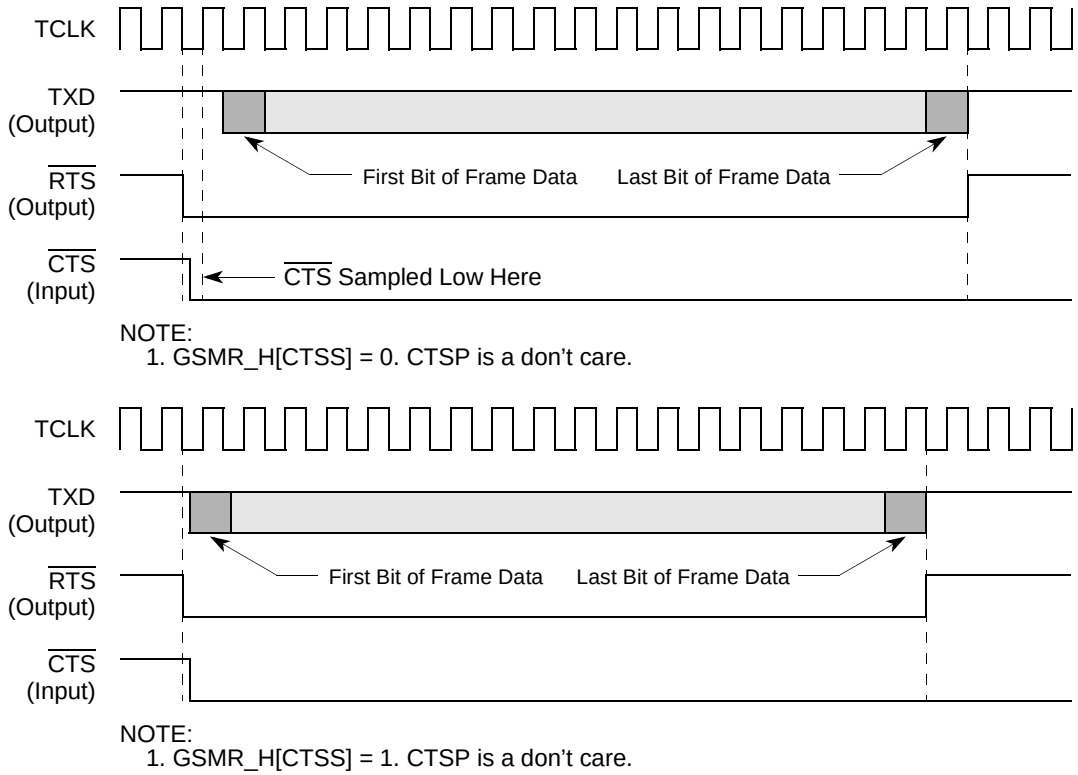


Figure 27-10. Output Delay from $\overline{\text{CTS}}$ Asserted for Synchronous Protocols

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.

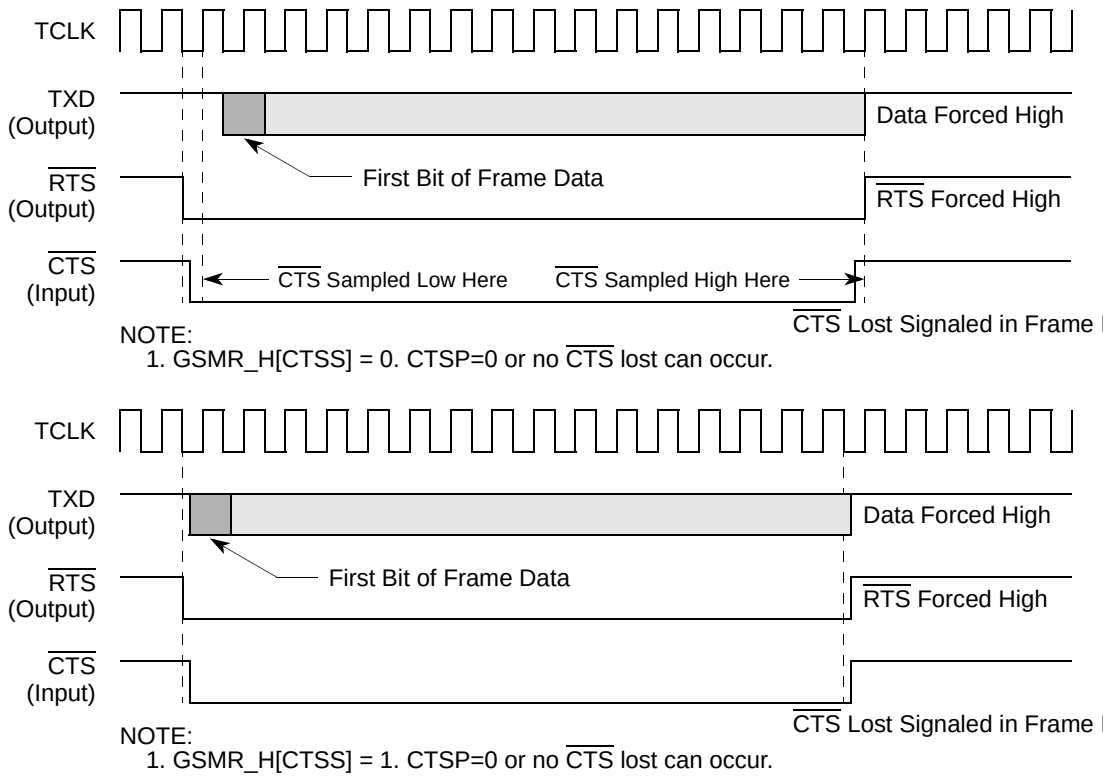


Figure 27-11. $\overline{\text{CTS}}$ Lost in Synchronous Protocols

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.

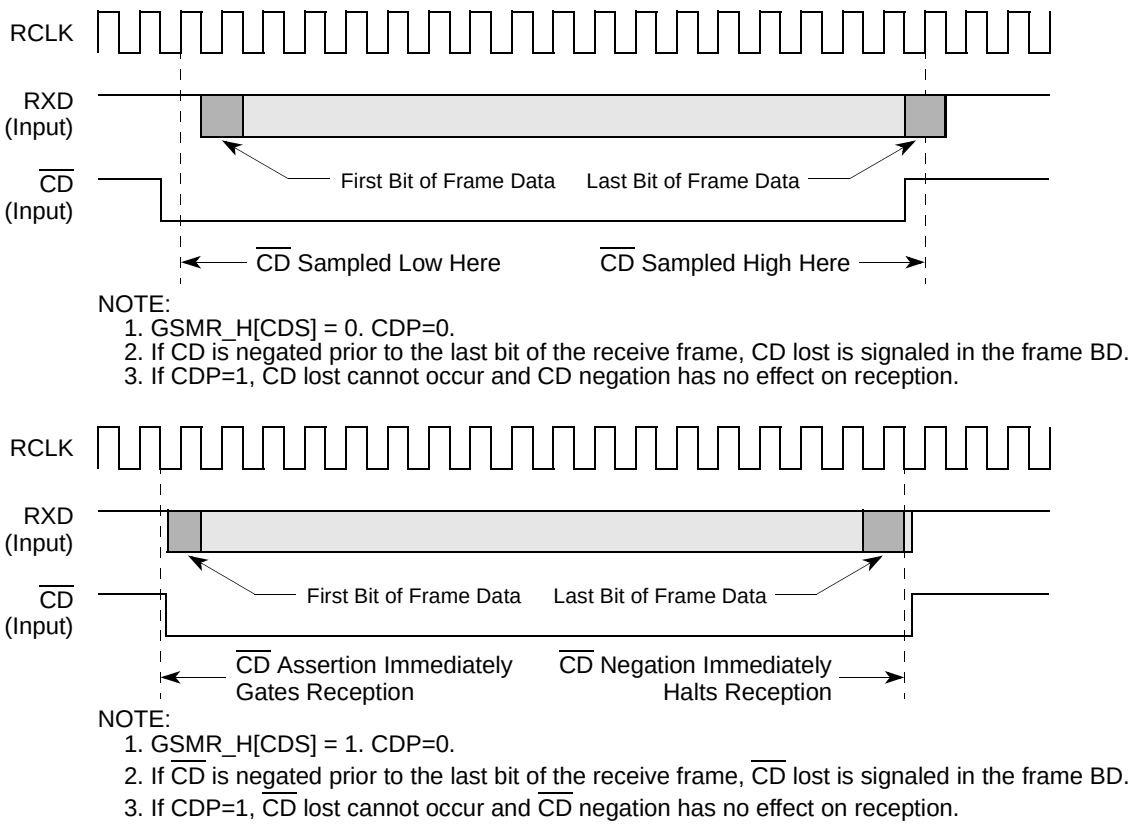


Figure 27-12. Using $\overline{\text{CD}}$ to Control Synchronous Protocol Reception

If $\overline{\text{CD}}$ is programmed to envelope the data, it must remain asserted during frame transmission or a $\overline{\text{CD}}$ lost error occurs. Negation of $\overline{\text{CD}}$ terminates reception. If GSMR_H[CDS] is zero, $\overline{\text{CD}}$ must be sampled by the SCC before a $\overline{\text{CD}}$ lost error is recognized; otherwise, the negation of $\overline{\text{CD}}$ immediately causes the $\overline{\text{CD}}$ lost condition.

If GSMR_H[CDS] is set, all $\overline{\text{CD}}$ transitions must occur while the Rx clock is low.

27.4.5.2 Asynchronous Protocols

In asynchronous protocols, $\overline{\text{RTS}}$ is asserted when SCC data is loaded into the Tx FIFO and a falling Tx clock occurs. $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ 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 $\overline{\text{CTS}}$ flow control as described in Chapter 28, "SCC UART Mode."

- If $\overline{\text{CTS}}$ is already asserted when $\overline{\text{RTS}}$ is asserted, transmission begins in two additional bit times.
- If $\overline{\text{CTS}}$ is not already asserted when $\overline{\text{RTS}}$ is asserted and GSMR_H[CTSS] = 0, transmission begins in three additional bit times.
- If $\overline{\text{CTS}}$ is not already asserted when $\overline{\text{RTS}}$ 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.

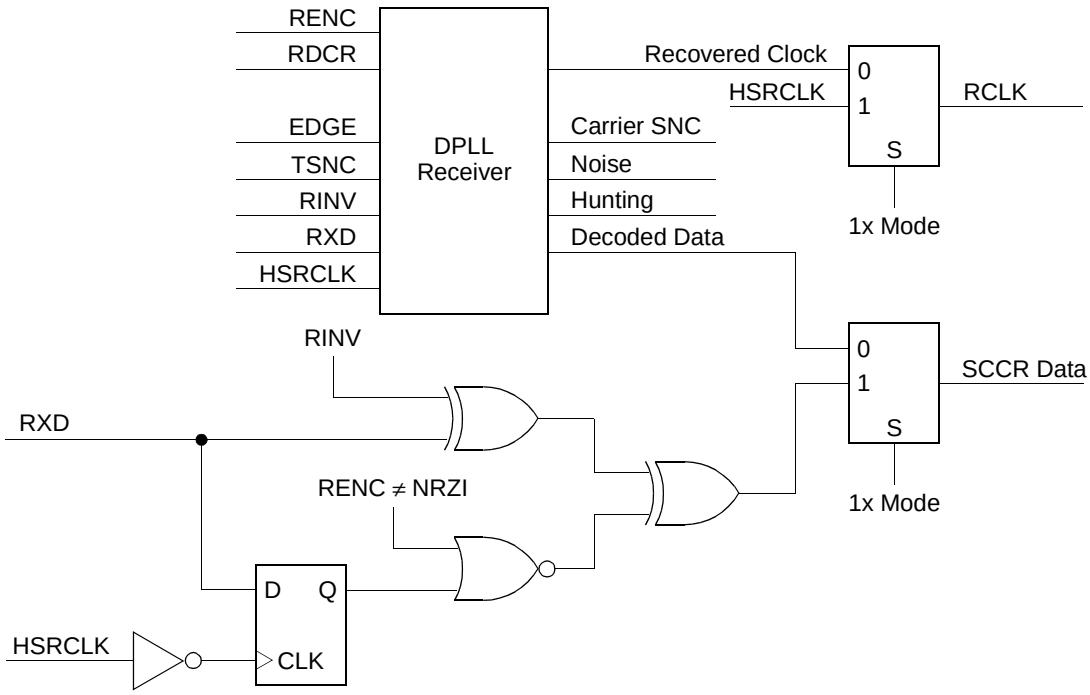


Figure 27-13. DPLL Receiver Block Diagram

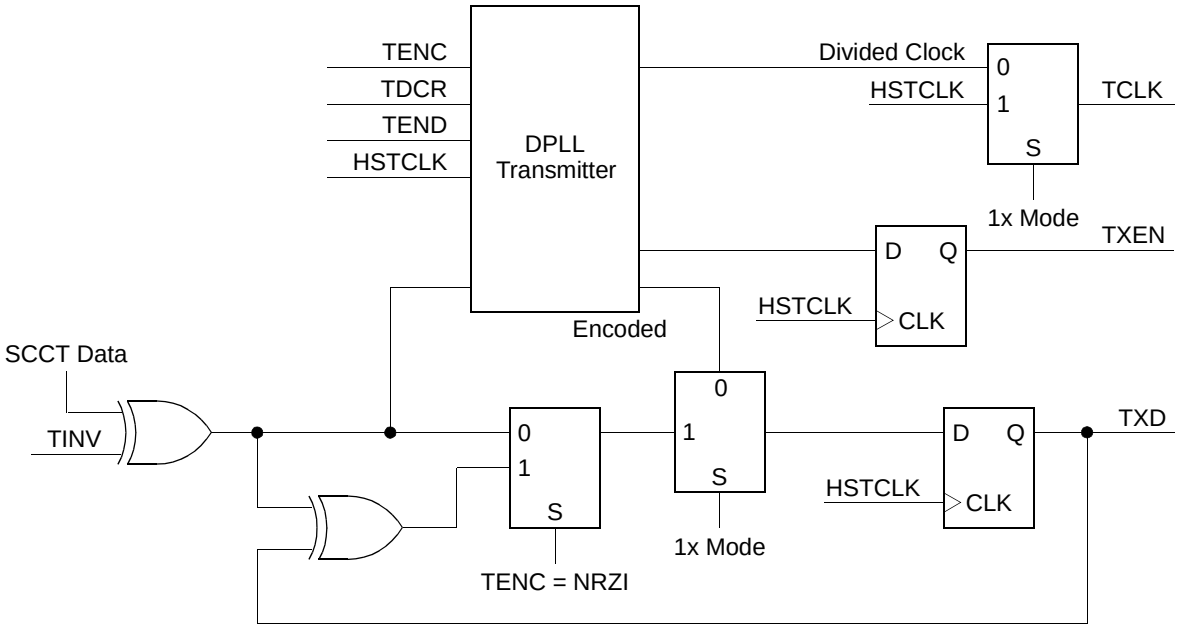


Figure 27-14. DPLL Transmitter Block Diagram

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.

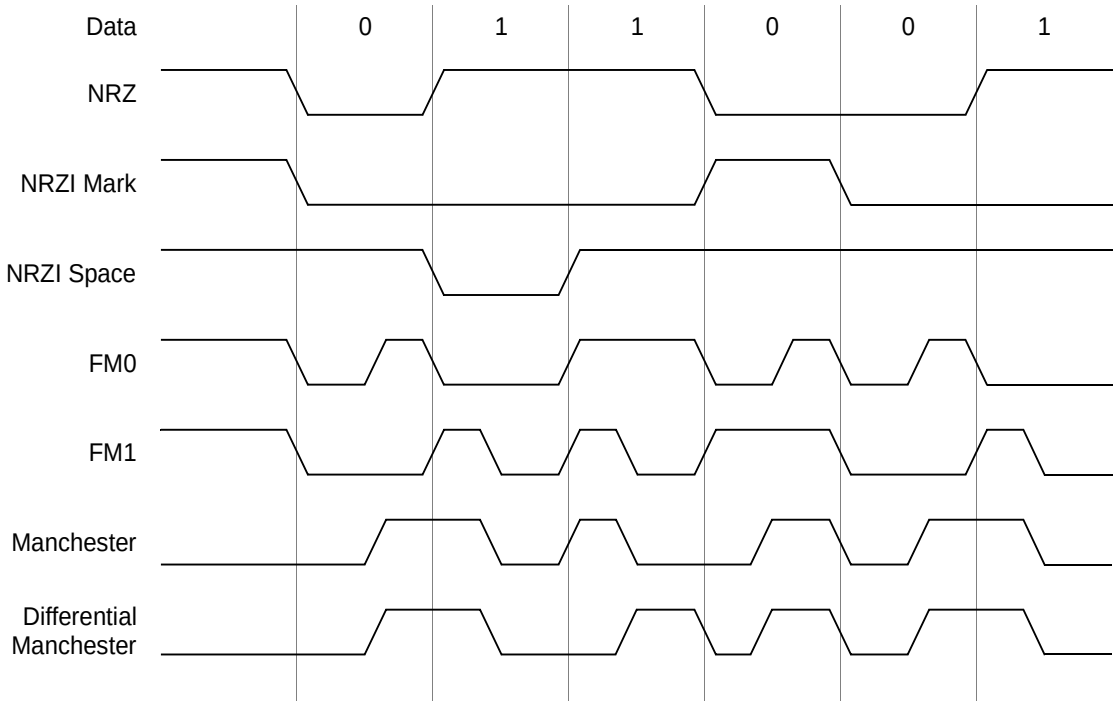


Figure 27-15. DPLL Encoding Examples

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 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.

27.4.7.1 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.

27.4.7.2 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].

27.4.7.3 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.

27.4.7.4 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].

27.4.7.5 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.

27.4.8 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.

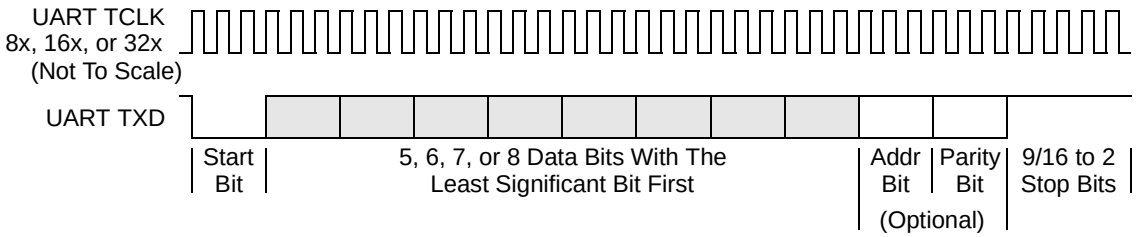


Figure 28-1. UART Character Format

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 wakeup 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[2]	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.
0x40	**FRMEC**		
0x42	**NOSEC**		
0x44	**BRKEC**		
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	
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	

Table 28-1. UART-Specific SCC Parameter RAM Memory Map (continued)

Offset[1]	Name[2]	Width	Description
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 = 0b001xxxxxxxxxxxxx indicates two zeros; 0b1xxxxxxxxxxxxxxx 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.”

2 **Boldfaced** entries must be initialized by the user.

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 multi buffer, 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.
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.

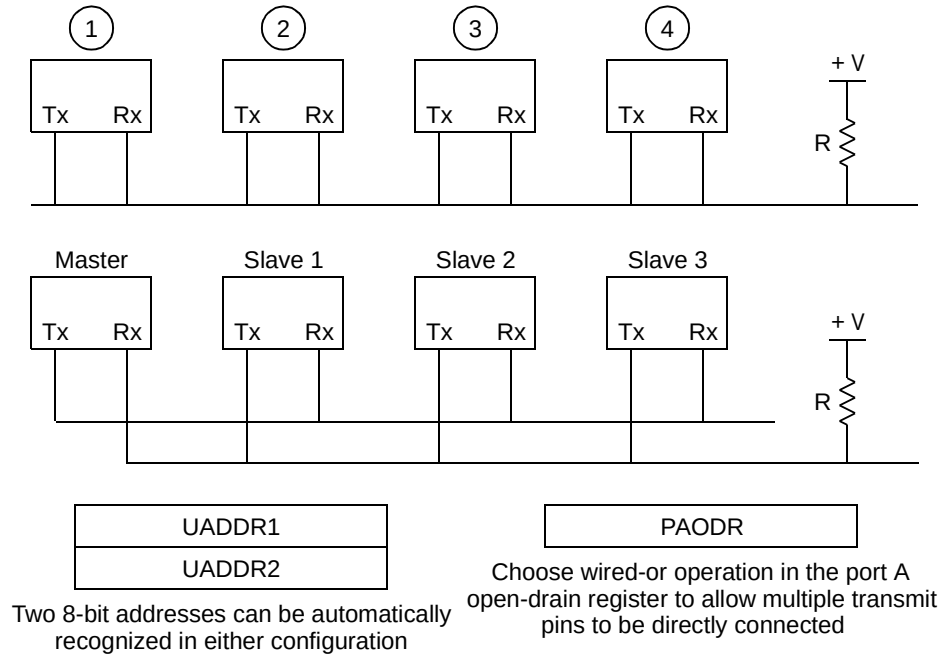


Figure 28-2. Two UART Multidrop Configurations

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[1]	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.

Table 28-4. Control Character Table, RCCM, and RCCR Descriptions (continued)

Offset	Bits	Name[1]	Description
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.

1 **Boldfaced** entries must be initialized by the user.

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.

	0	1	2	3	4	5	6	7	8	15
Field	—		**REA**	**I**	CT	—		**A**	**CHARSEND**	
Reset	0000_0000_0000_0000									
R/W	R/W									
Addr	SCC base + 0x4E									

Figure 28-4. Transmit Out-of-Sequence Register (TOSEQ)

Table 28-5 describes TOSEQ fields.

Table 28-5. TOSEQ Field Descriptions

Bit	Name[1]	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.
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.

1 **Boldfaced** entries must be initialized by the user.

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[1]	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.

Table 28-6. DSR Fields Descriptions (continued)

Bit	Name[1]	Description
5–6	—	0b11
7–8	—	0b00
9–14	—	0b111111
15	—	0b0

1 **Boldfaced** entries must be initialized by the user.

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.

Table 28-8. Reception Errors (continued)

Error	Description
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.
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.

Table 28-9. PSMR UART Field Descriptions (continued)

Bit	Name	Description
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.

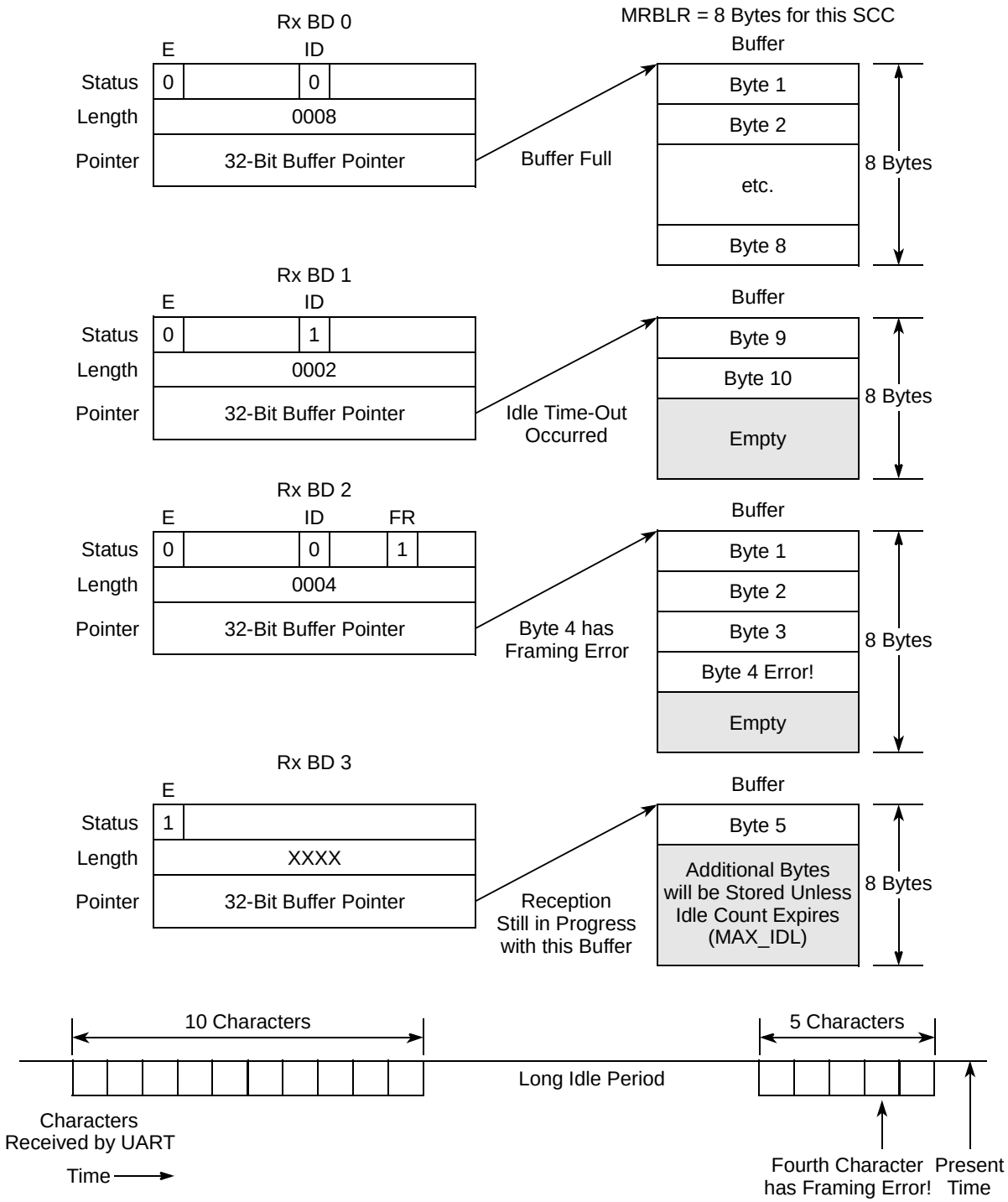


Figure 28-7. SCC UART Receiving using RxBDs

Figure 28-8 shows the SCC UART RxBD.

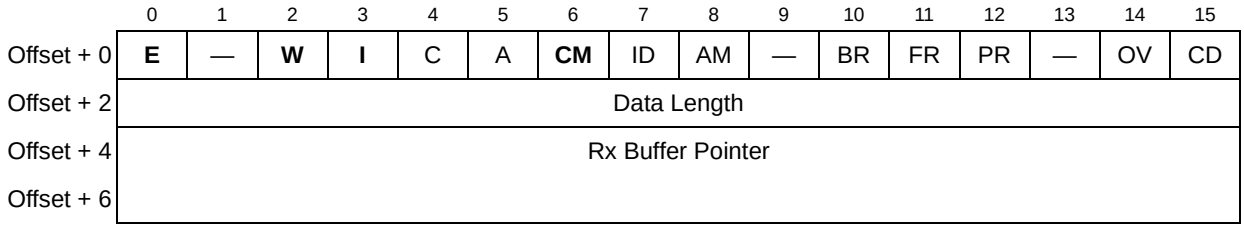


Figure 28-8. SCC UART Receive Buffer Descriptor (RxBD)

Table 28-10 describes RxBD status and control fields.

Table 28-10. SCC UART RxBD Status and Control Field Descriptions

Bits	Name[1]	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 programmable 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.

Table 28-10. SCC UART RxBD Status and Control Field Descriptions (continued)

Bits	Name[1]	Description
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.
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.

1 **Boldfaced** entries must be initialized by the user.

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.

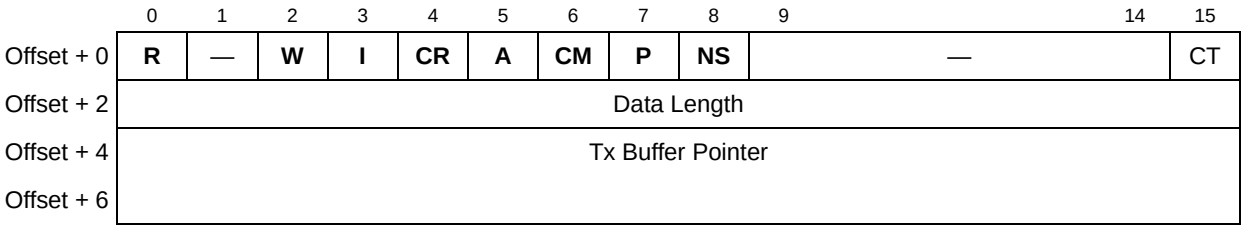


Figure 28-9. SCC UART Transmit Buffer Descriptor (TxBD)

Table 28-11 describes TxBD status and control fields.

Table 28-11. SCC UART TxBD Status and Control Field Descriptions

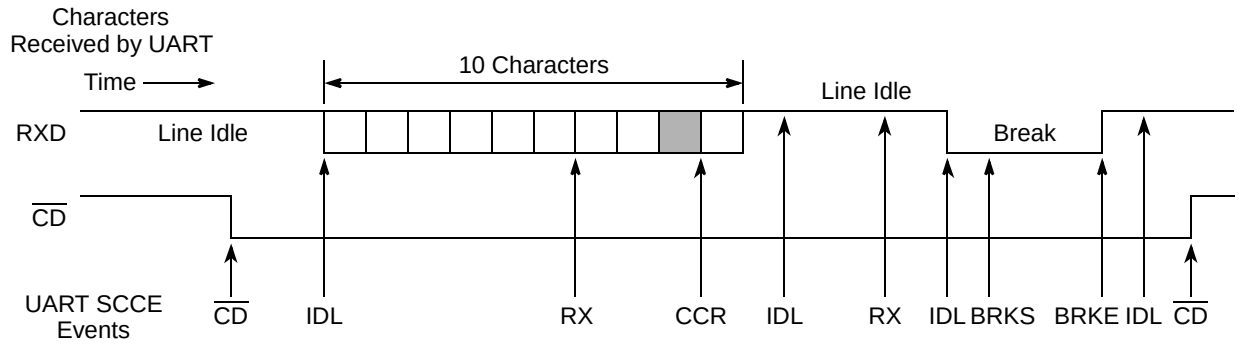
Bit	Name[1]	Description
0	**R**	Ready. 0 The buffer is not ready. This BD and buffer can be modified. The CPM automatically clears R after the buffer is sent or an error occurs. 1 The user-prepared buffer is waiting to begin transmission or is being transmitted. Do not modify the BD once R is set.
1	—	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.
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.

[1] **Boldfaced** entries must be initialized by the user.

The data length and buffer pointer fields are described in Section 27.3, “SCC Buffer Descriptors (BDs).”

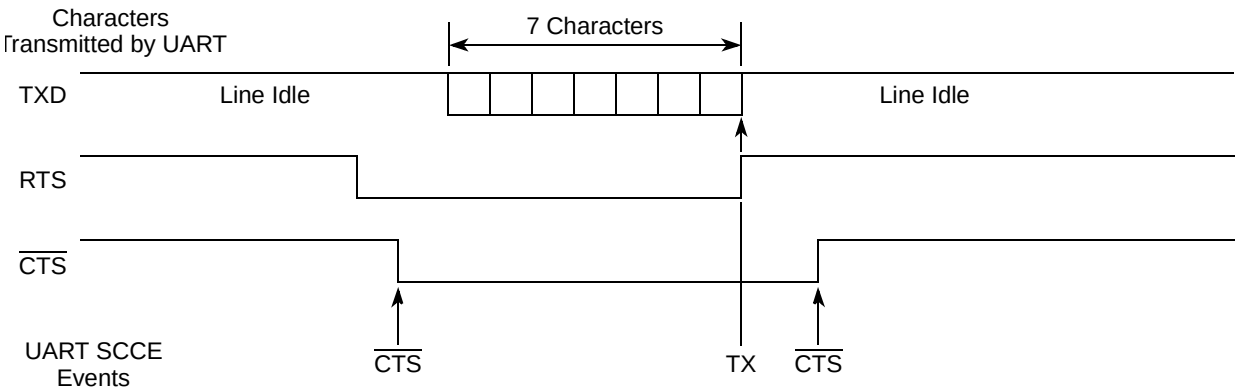
28.19 SCC UART Event Register (SCCE) and Mask Register (SCCM)

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	—						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
0–5	—	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.

Opening Flag	Address	Control	Information (Optional)	CRC	Closing Flag
8 bits	16 bits	8 bits	8*n* bits	16 bits	8 bits

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 in Table 29-1.

Table 29-1. HDLC-Specific SCC Parameter RAM Memory Map

Offset[1]	Name[2]	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."
2 **Boldfaced** entries must be initialized by the user.

Figure 29-2 shows 16- and 8-bit address recognition.

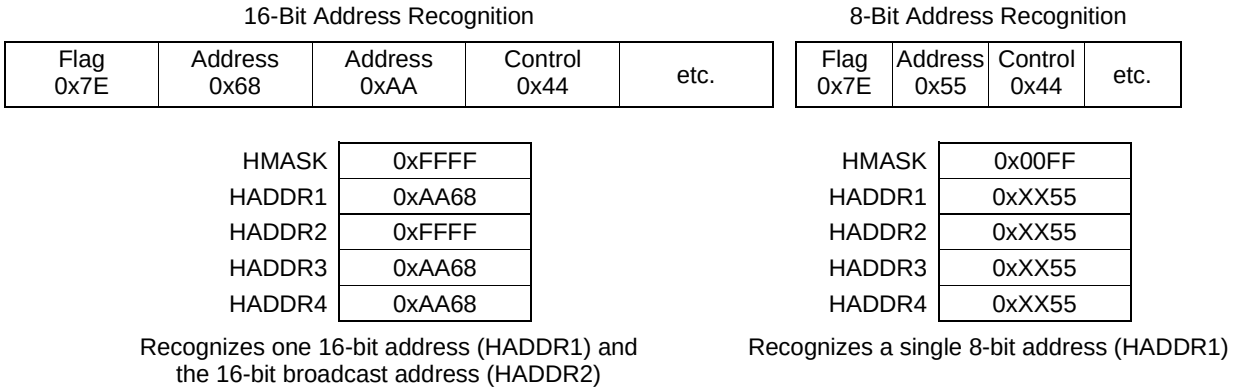


Figure 29-2. HDLC Address Recognition

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 multi buffer 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.

Table 29-5 describes reception errors.

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.
$\overline{\text{CD}}$ 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[1]	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.

Table 29-7. SCC HDLC RxBD Status and Control Field Descriptions (continued)

Bits	Name[1]	Description
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.
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 $\overline{\text{CD}}$ is negated during frame reception.

1 **Boldfaced** entries must be initialized by the user.

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.

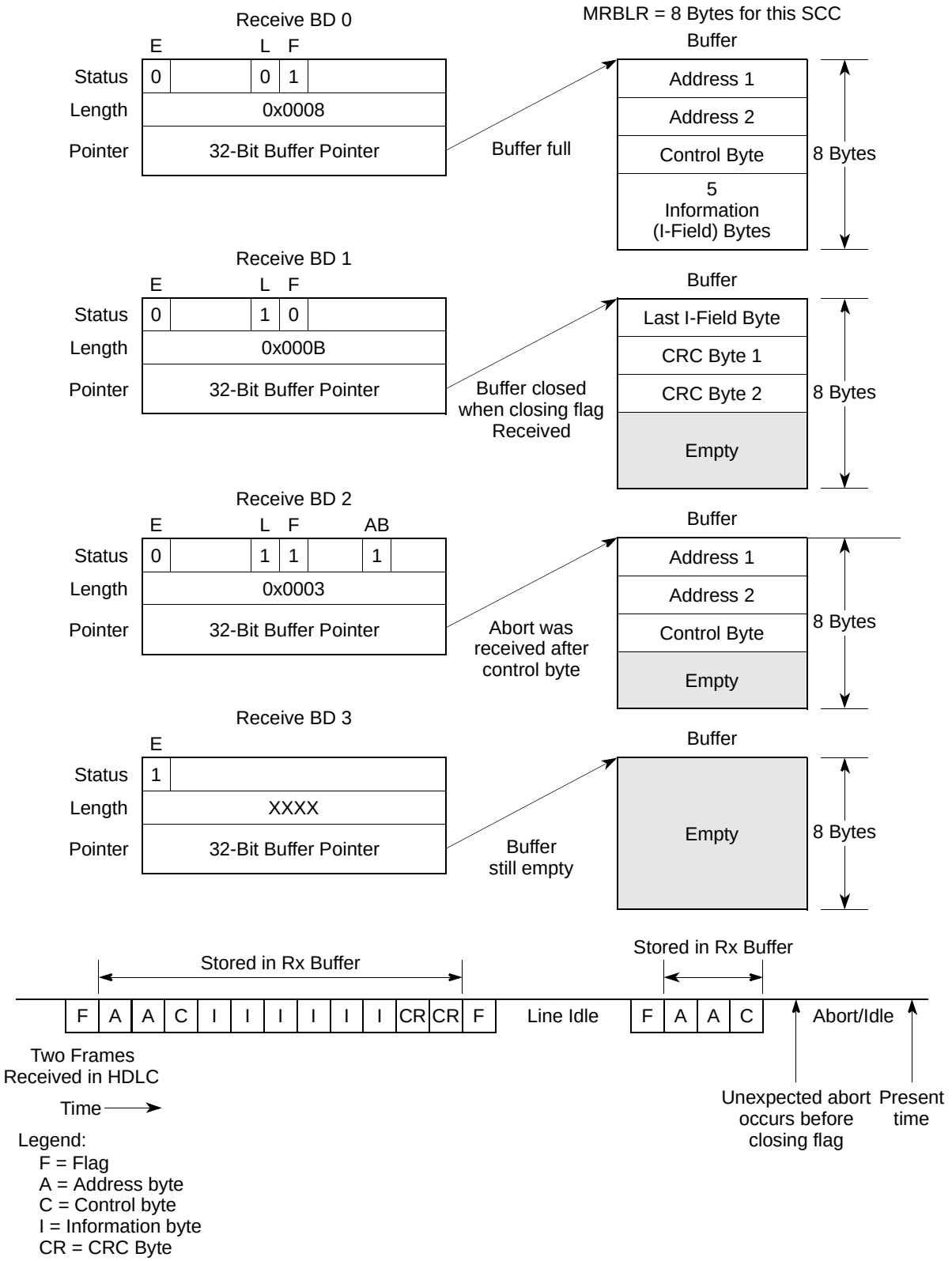


Figure 29-5. SCC HDLC Receiving Using RxBDs

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.

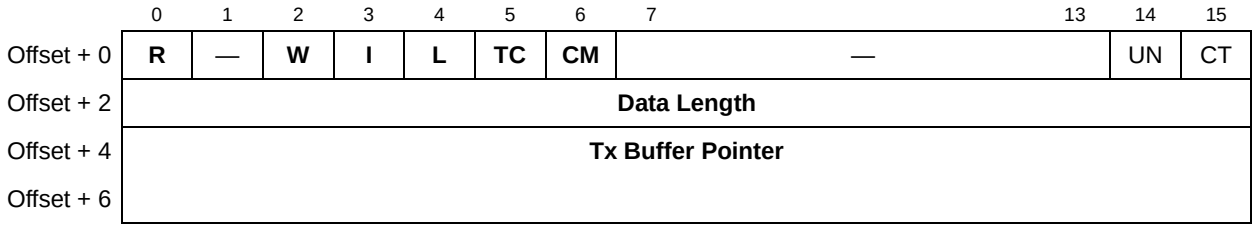


Figure 29-6. SCC HDLC Transmit Buffer Descriptor (TxBD)

Table 29-8 describes HDLC TxBD status and control fields.

Table 29-8. SCC HDLC TxBD Status and Control Field Descriptions

Bits	Name[1]	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.

[1] **Boldfaced** entries must be initialized by the user.

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.

	0–4	5	6	7	8	9–10	11	12	13	14	15
Field	—	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
0–4	—	Reserved, should be cleared.
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 $\overline{\text{CD}}$ 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 ($\overline{\text{CTS}}$ 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.

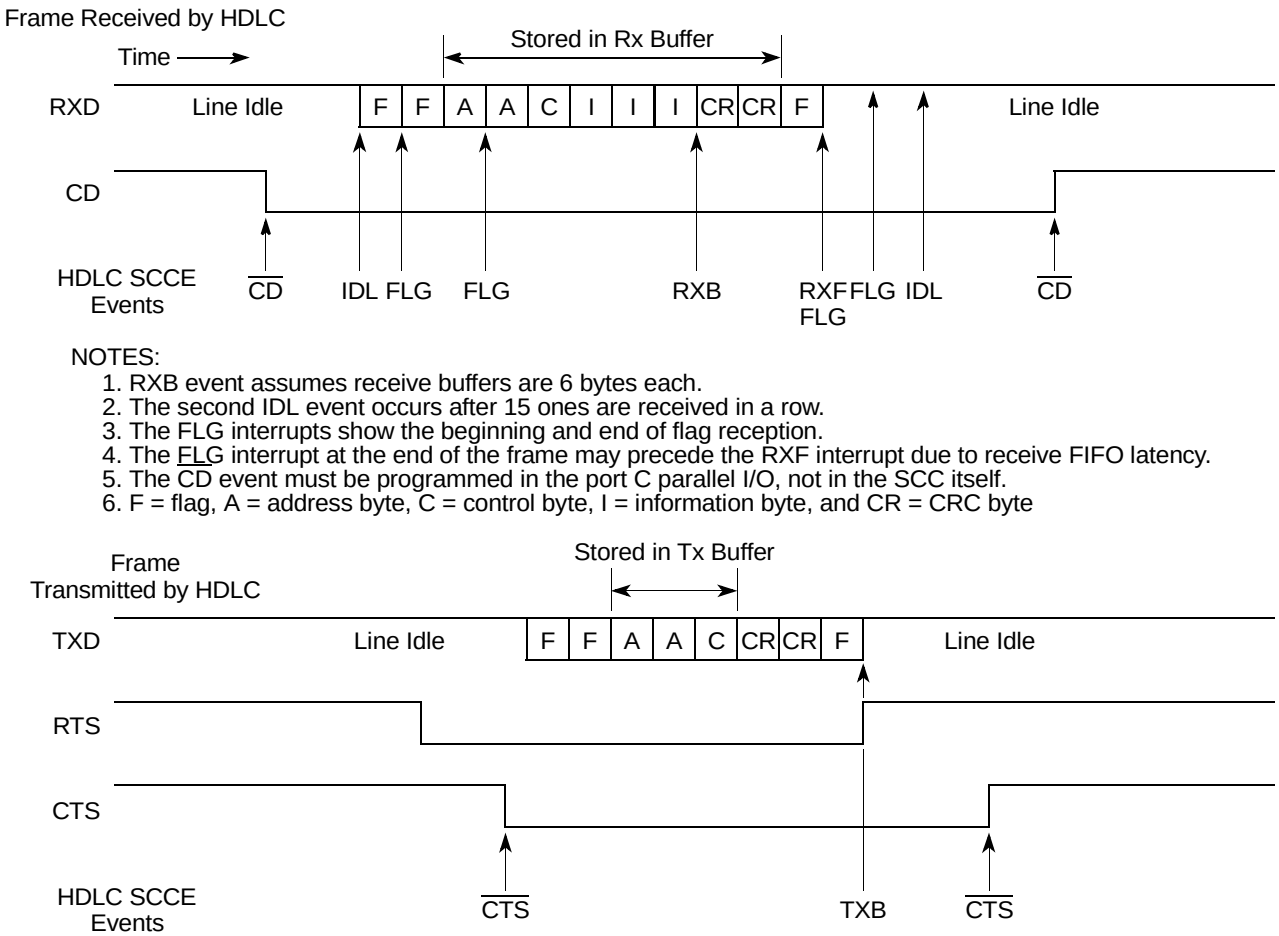


Figure 29-8. SCC HDLC Interrupt Event Example

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 $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ 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.

29.13.1 SCC 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 CPM interrupt mask register low (SIMR_L) so the SCC2 can generate a system interrupt. Initialize CPM 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.

29.13.2 SCC 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.

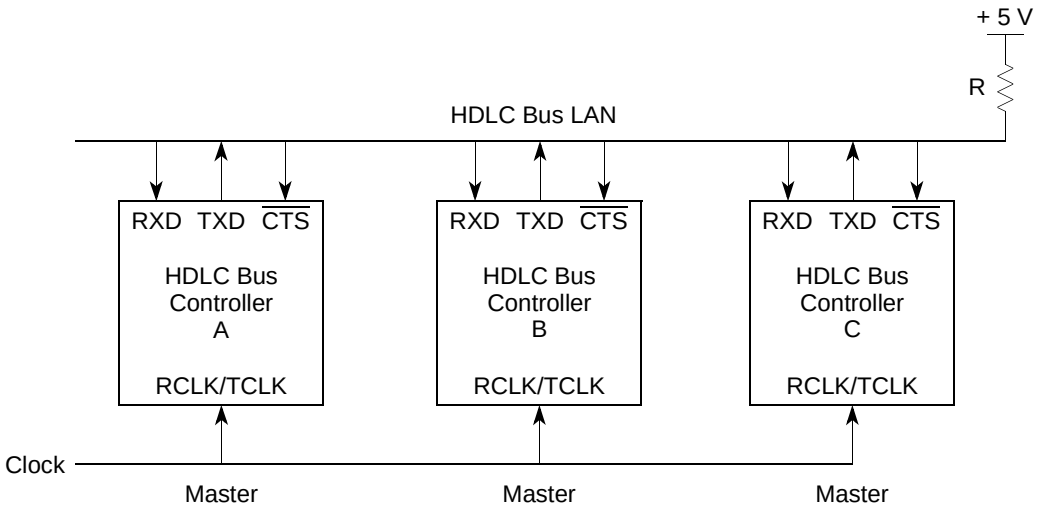
The HDLC bus differs from the I.430 and T1.605 standards as follows:

- 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

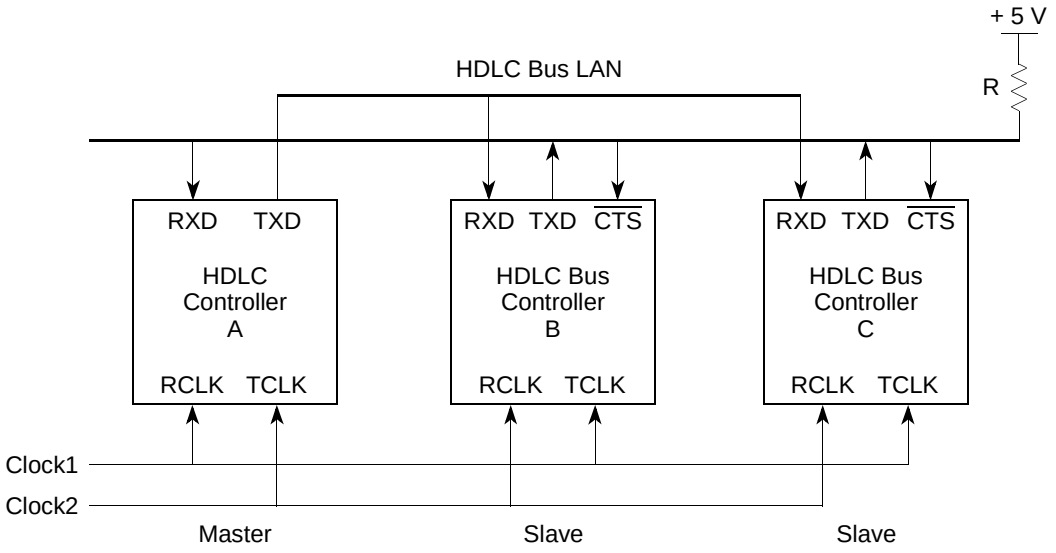
Figure 29-10 shows the most common HDLC bus LAN configuration, a multiple-master configuration. A station can transfer data to or from any other LAN station. Transmissions are half-duplex, which is typical in LANs.



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 Multiple-Master 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 $\overline{\text{CTS}}$ bus sample, transmission continues. However, if the received $\overline{\text{CTS}}$ 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 $\overline{\text{CTS}}$ is used to detect collisions.

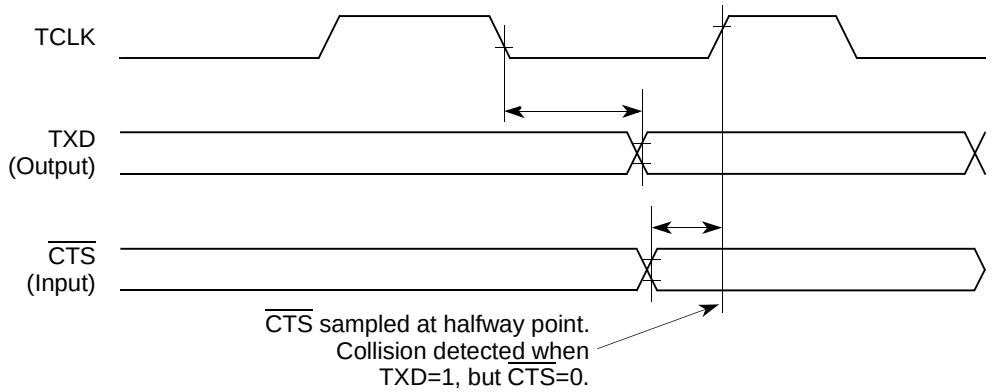


Figure 29-12. Detecting an HDLC Bus Collision

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.

29.14.3 Increasing 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.

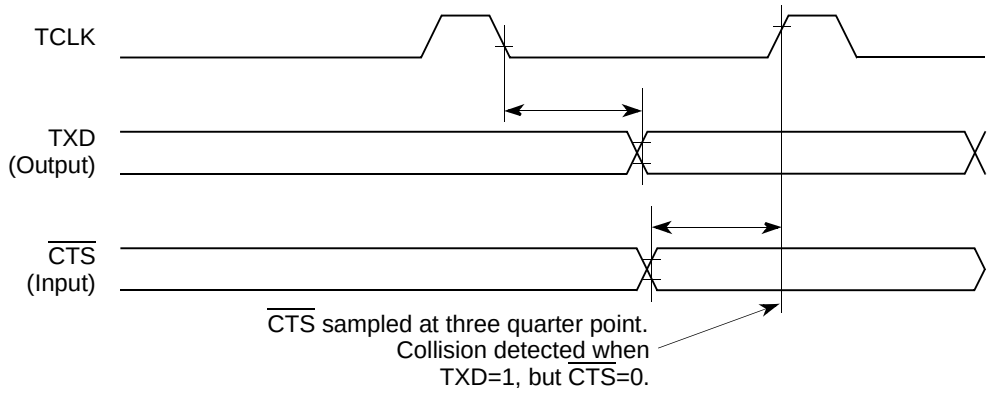
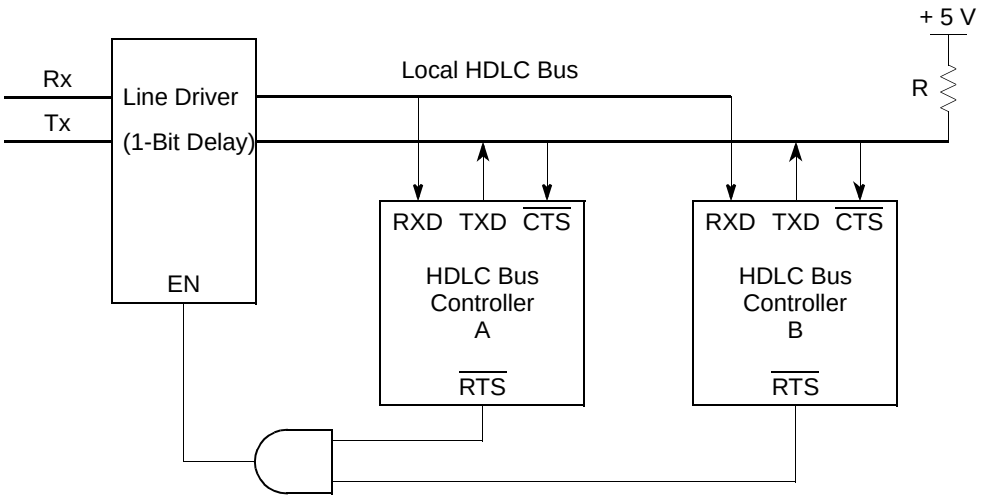


Figure 29-13. Nonsymmetrical Tx Clock Duty Cycle for Increased Performance

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.

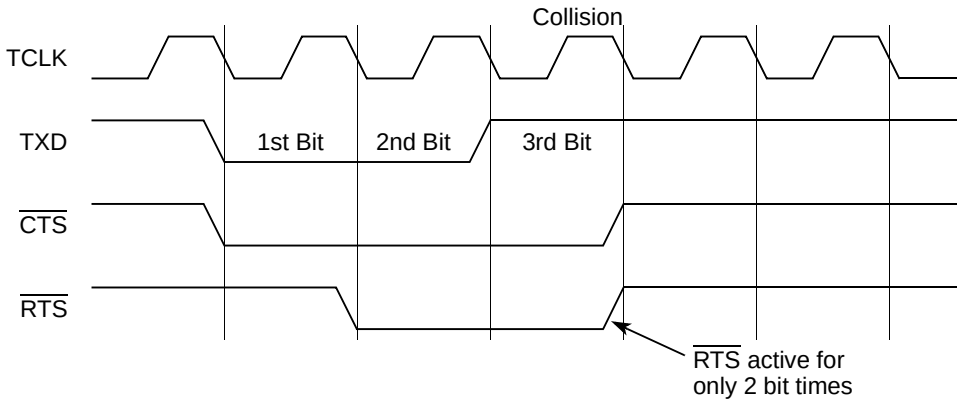
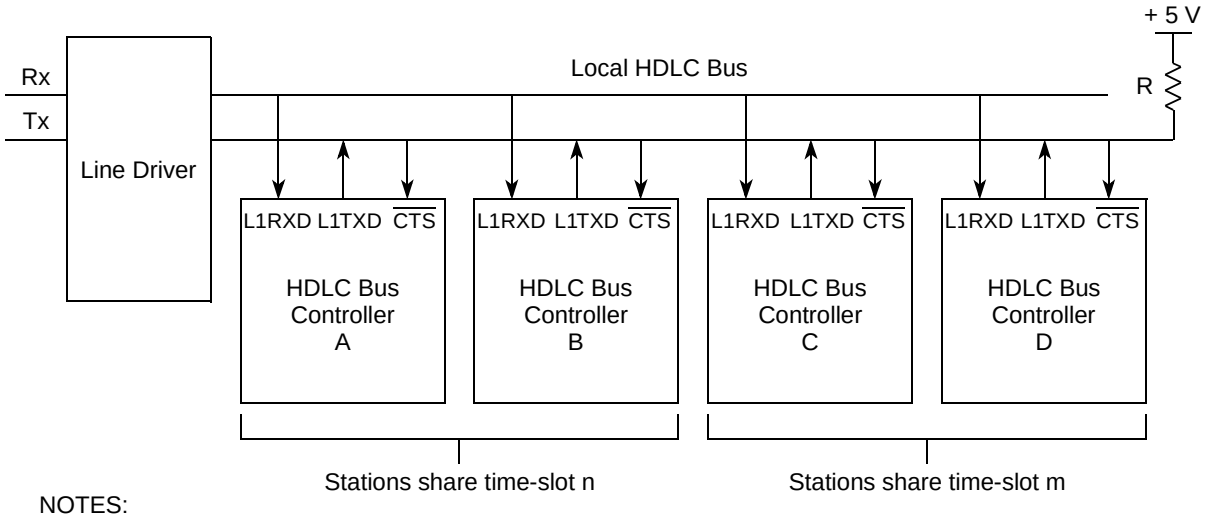


Figure 29-15. Delayed RTS Mode

29.14.5 Using 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. The three classes of BISYNC frames, shown in Figure 30-1, are transparent, nontransparent with header, and nontransparent without header. 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.

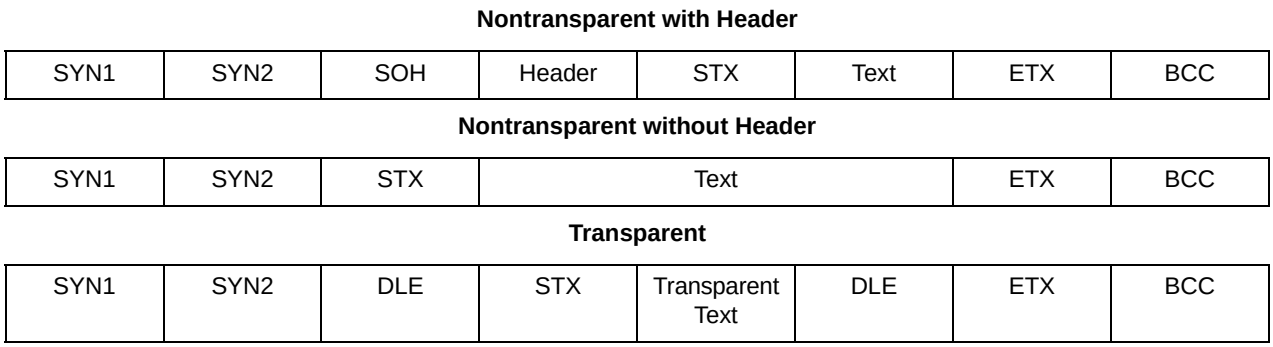


Figure 30-1. Classes of BISYNC Frames

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 CR 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 shown in Table 30-1.

Table 30-1. SCC BISYNC Parameter RAM Memory Map

Offset [1]	Name[2]	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.”
2 **Boldfaced** entries must be initialized by the user.

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.

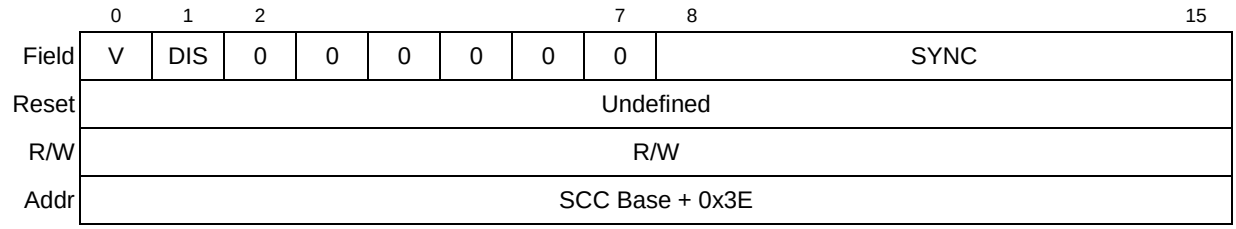


Figure 30-3. BISYNC SYNC (BSYNC)

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.

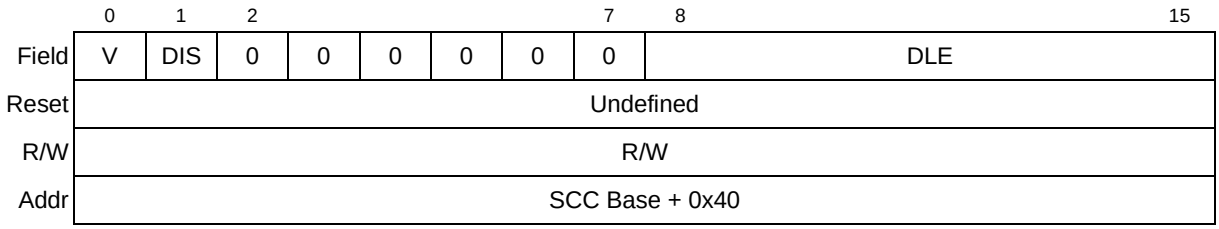


Figure 30-4. BISYNC DLE (BDLE)

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.

Table 30-10. PSMR Field Descriptions (continued)

Bits	Name	Description
10–11	—	Reserved, should be cleared.
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 shows the SCC BISYNC RxBD.

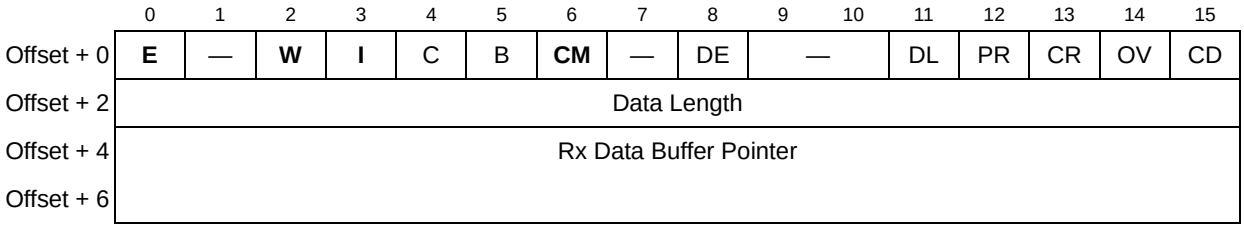


Figure 30-6. SCC BISYNC RxBD

Table 30-11 describes SCC BISYNC RxBD status and control fields.

Table 30-11. SCC BISYNC RxBD Status and Control Field Descriptions

Bits	Name[1]	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	C	Control character. The last byte in the buffer is a user-defined control character. 0 The last byte of this buffer does not contain a control character. 1 The last byte of this buffer contains a control character.
5	B	BCS received. The last bytes in the buffer contain the received BCS. 0 This buffer does not contain the BCS. 1 This buffer contains the BCS. A control character may also reside one byte prior to this BCS.
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	DL	DLE follow character error. While in transparent mode, a DLE character was received, and the next character was not DLE, SYNC, or a valid entry in the control characters table.
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	BCS error. Updated every time a byte is written to the buffer. The CR bit includes the calculation for the current byte. By clearing PSMR[RCBS] within eight serial clocks, the user can exclude the current character from the message BCS calculation. The data length field may be read to determine the current character's position.
14	OV	Overrun. Set when a receiver overrun occurs during frame reception.
15	CD	Carrier detect lost. Indicates when the carrier detect signal, $\overline{\text{CD}}$, is negated during frame reception.

[1] **Boldfaced** entries must be initialized by the user.

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.

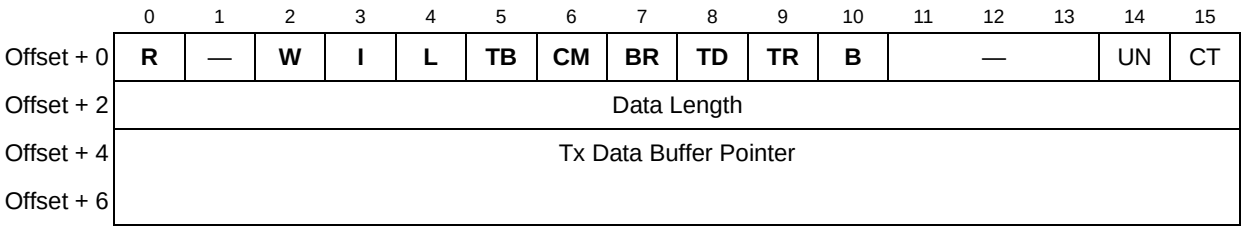


Figure 30-7. SCC BISYNC Transmit BD (TxBD)

Table 30-12 describes SCC BISYNC TxBD status and control fields.

Table 30-12. SCC BISYNC TxBD Status and Control Field Descriptions

Bits	Name[1]	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.

Table 30-12. SCC BISYNC TxBD Status and Control Field Descriptions (continued)

Bits	Name[1]	Description
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.
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.

[1] **Boldfaced** entries must be initialized by the user.

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.

	0–4	5	6–7	8	9–10	11	12	13	14	15
Field	—	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
0–4	—	Reserved, should be cleared.
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	1	2	3	4	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 CPM interrupt mask register low (SIMR_L) so the SCC2 can generate a system interrupt. Initialize CPM 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 with 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

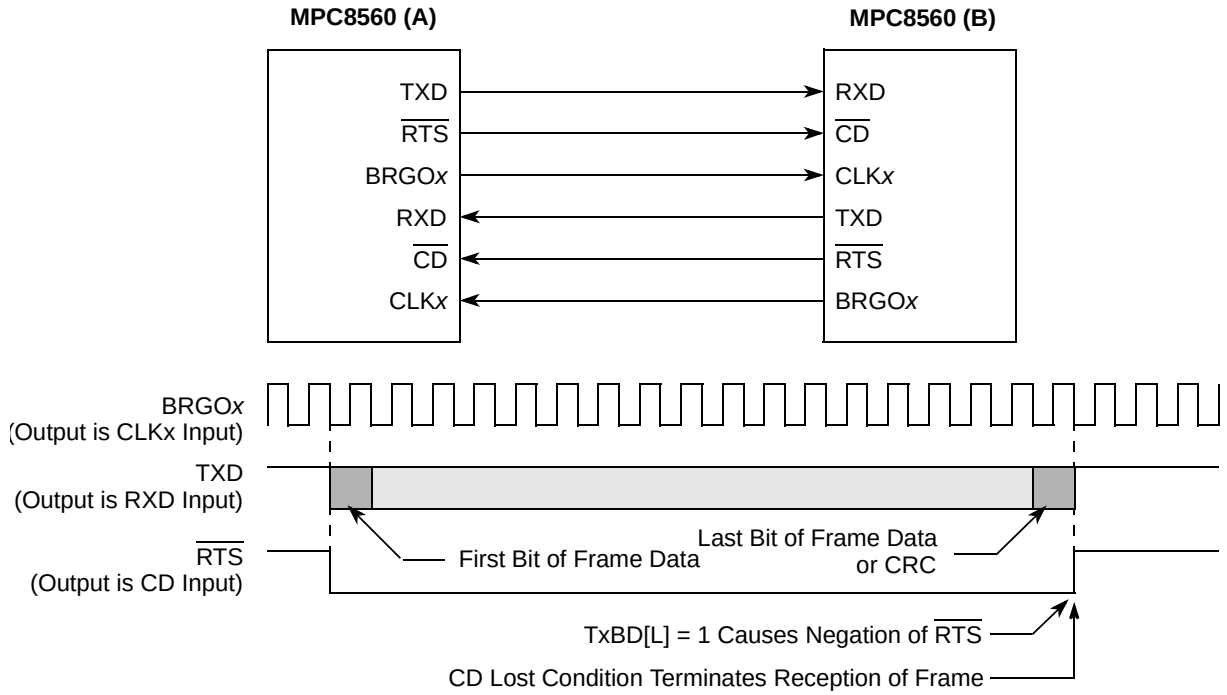
If GSMR_H[SYNL] is 0b00, the transmitter uses $\overline{\text{CTS}}$ and the receiver uses $\overline{\text{CD}}$ to begin the sequence. These signals share two options—pulsing and sampling.

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 31-2.

Table 31-2. SCC Transparent Parameter RAM Memory Map

Offset[1]	Name	Width	Description
0x30	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).
0x34	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.

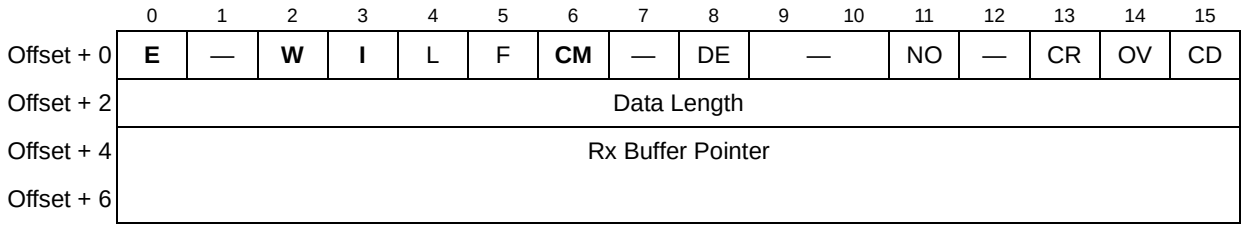


Figure 31-2. SCC Transparent Receive Buffer Descriptor (RxBD)

Table 31-7 describes RxBD status and control fields.

Table 31-7. SCC Transparent RxBD Status and Control Field Descriptions

Bits	Name[1]	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.

1 **Boldfaced** entries must be initialized by the user.

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.

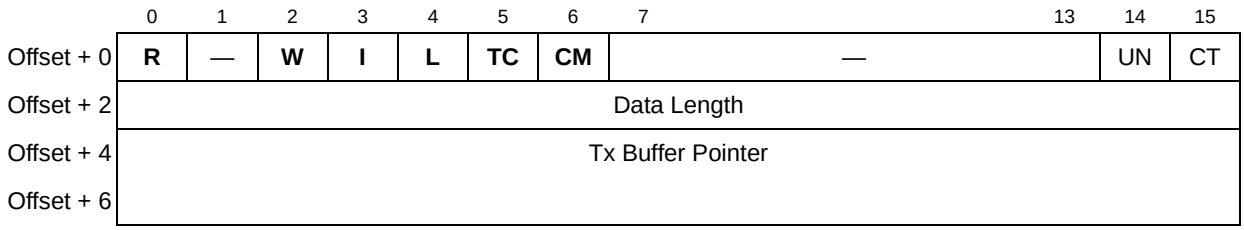


Figure 31-3. SCC Transparent Transmit Buffer Descriptor (TxBD)

Figure 31-8 describes SCC Transparent TxBD status and control fields.

Table 31-8. SCC Transparent TxBD Status and Control Field Descriptions

Bits	Name[1]	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.

Table 31-8. SCC Transparent TxBD Status and Control Field Descriptions (continued)

Bits	Name[1]	Description
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.
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.

[1] **Boldfaced** entries must be initialized by the user.

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 CPM interrupt controller. Figure 31-4 shows the event and mask registers.

	0–4	5	6–7	8	9–10	11	12	13	14	15
Field	—	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
0–4	—	Reserved, should be cleared.
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 $\overline{\text{CD}}$ 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 $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ 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 CPM interrupt mask register low (SIMR_L) so SCC2 can generate a system interrupt. Initialize CPM 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.

Sync Sequence	HDLC Flags	Destination Address	Source Address	Control Byte	Data (Optional)	CRC-16	Closing Flag	Abort Sequence
> 3 Bits	2 or More Bytes	1 Byte	1 Byte	1 Byte	0-600 Bytes	2 Bytes	1 Byte	12–18 Ones

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.

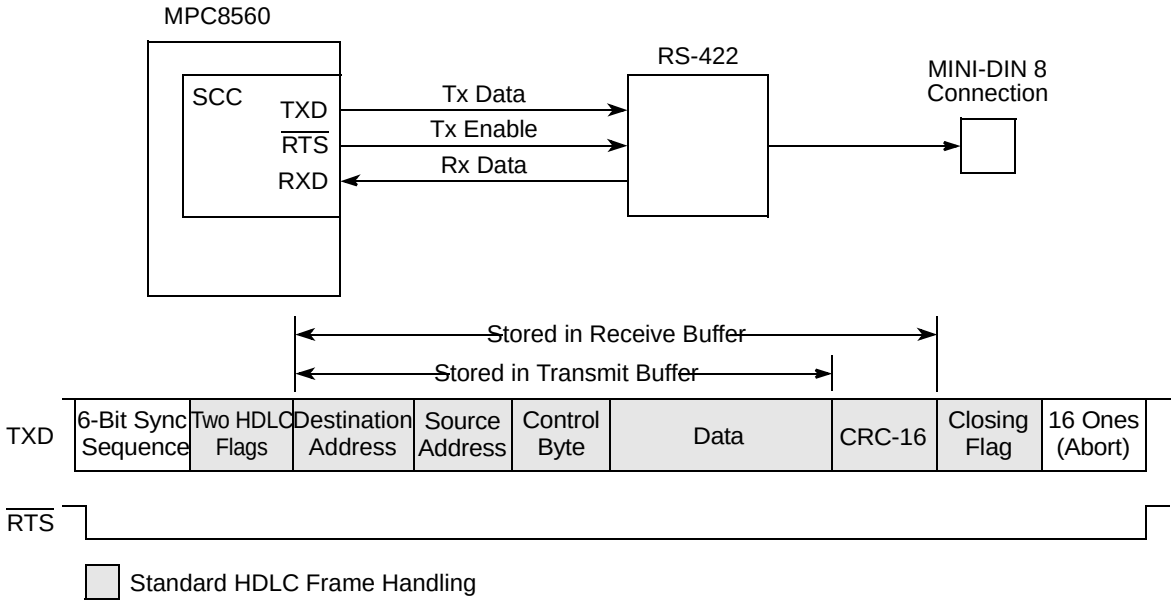


Figure 32-2. Connecting the MPC8560 to LocalTalk

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 $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ grounded or configured for parallel I/O. This causes $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ 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 20-14..
- 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).”

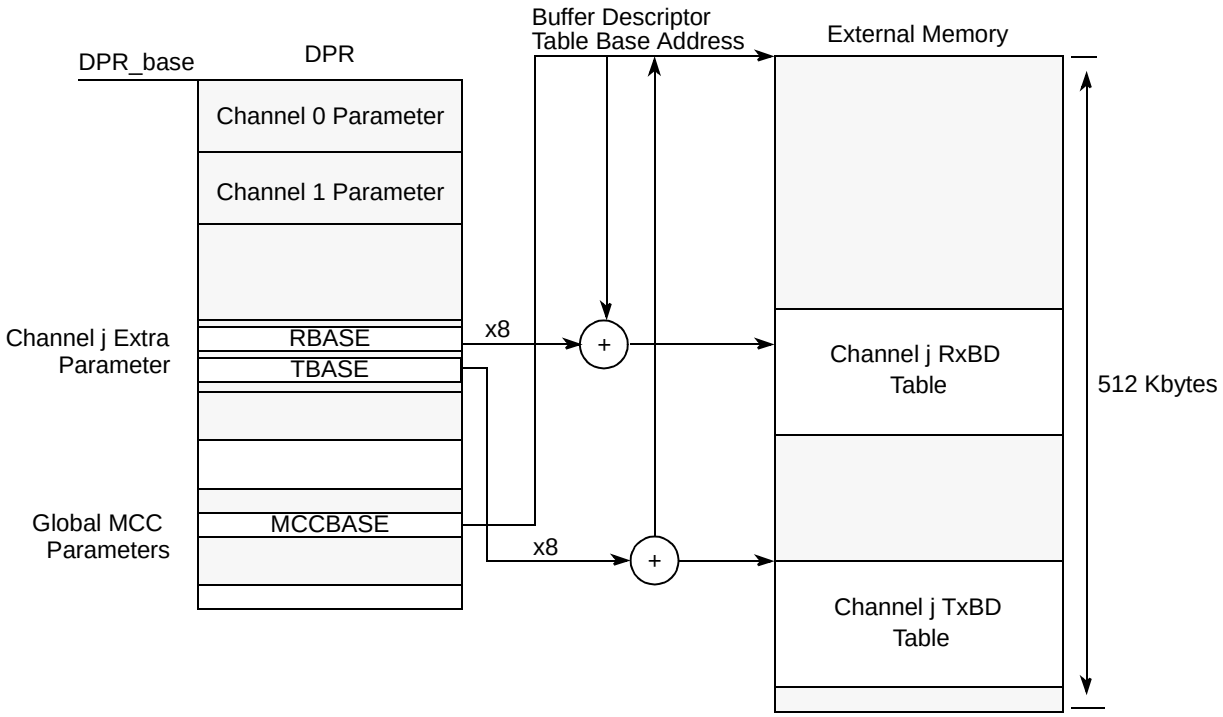


Figure 33-1. BD Structure for One MCC

33.4 Global MCC Parameters

The global MCC parameters are described in Table 33-1.

Table 33-1. Global MCC Parameters

Offset[1]	Name[2]	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[2]	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.

2 **Boldfaced** entries must be initialized by the user.

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[2]	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

2 **Boldfaced** entries must be initialized by the user.

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	0x5	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	0x8	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	0x0	1	0	Regular Channel
1	0	0	0x1	0x0	1	0	Super Channel 1
1	0	0	0x2	0x0	1	0	Super Channel 2
1	0	0	0x2	0x0	1	0	Super Channel 2
1	0	0	0x2	0x0	1	0	Super Channel 2
1	0	0	0x5	0x0	1	0	Regular Channel
1	0	0	0x1	0x0	1	0	Super Channel 1
1	0	0	0x1	0x0	1	0	Super Channel 1
1	0	0	0x8	0x0	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[2]	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).”

Table 33-3. Channel-Specific Parameters for HDLC (continued)

Offset[1]	Name[2]	Width	Description
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
2 **Boldfaced** entries must be initialized by the user.

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.

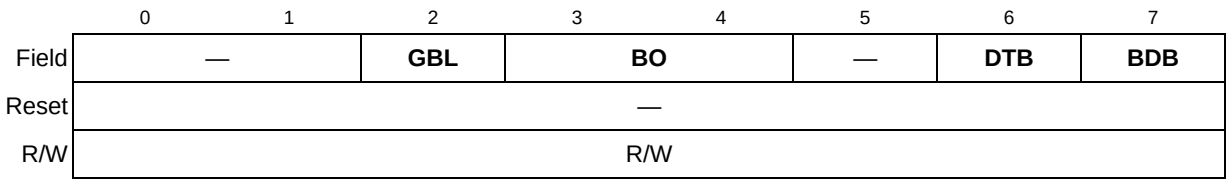


Figure 33-6. TSTATE High Byte

TSTATE high-byte fields are described in Table 33-4.

Table 33-4. TSTATE High-Byte Field Descriptions

Bits	Name[1]	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

1 **Boldfaced** entries must be initialized by the user.

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 [1]	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, that is, 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	—	This bit must be cleared.
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.

Table 33-5. CHAMR Field Descriptions (continued)

Bits	Name [1]	Description
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

1 **Boldfaced** entries must be initialized by the user.

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[1]	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).

Table 33-6. RSTATE High-Byte Field Descriptions (continued)

Bits	Name[1]	Description
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).

1 **Boldfaced** entries must be initialized by the user.

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[2]	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.”
0x1C	—	Word	Reserved

Table 33-7. Channel-Specific Parameters for Transparent Operation (continued)

<table>
<tr><th>Offset[1]</th><th>Name[2]</th><th>Width</th><th>Description</th></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.
[2] **Boldfaced** entries must be initialized by the user.

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[1]	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, that is, 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).

Table 33-8. CHAMR Field Descriptions—Transparent Mode (continued)

Bits	Name[1]	Description			
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).
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.			

1 **Boldfaced** entries must be initialized by the user.

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[2]	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)

Table 33-9. Channel-Specific Parameters for SS7 (continued)

Offset[1]	Name[2]	Width	Description
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 Section 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)
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 Section 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 Section 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).”

Table 33-9. Channel-Specific Parameters for SS7 (continued)

Offset[1]	Name[2]	Width	Description
0x48	**Mask1**	Word	Mask for SU filtering, bytes 0-3. See Section 33.9.4.4, “SU Filtering.”
0x4C	**Mask2**	Hword	Mask for SU filtering, byte 4. See Section 33.9.4.4, “SU Filtering.”
0x4E	**SS7_OPT**	Hword	SS7 configuration register. See Section 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 Section 33.9.4.4, “SU Filtering.”
0x5C	LRB2	Hword	Fifth byte of last received signal unit. Used by CP for SU filtering. See Section 33.9.4.4, “SU Filtering.”
0x5E	**T**	Hword	SUERM threshold value (user initialized to 64). See Section 33.9.2, “Signal Unit Error Monitor (SUERM).”
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).”
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, and so on) must be used and odd channel number must be left unused.

2 **Boldfaced** entries must be initialized by the user.

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 [1]	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 Section 33.12.1.1, “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, that is, 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 [1]	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.
26	**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).
27–28	**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
29–31	**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

[1] **Boldfaced** entries 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 has errors.
- 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.

	0–3	4	5	6	7	8	9	10	11	12–15
Field	—	AERM	SUERM_DIS	STD	SF_DIS	SU_FIL	SEN_FIS	O_ORN	O_ITUT	FISU_PAD
Reset										
R/W	R/W									

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[1]	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.

Table 33-12. SS7 Configuration Register Fields Description (continued)

Bits	Name[1]	Description
6	**STD**	Standard compliance 0 ITU-T/ANSI compliant 1 Japanese SS7 compliant
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 33.13.2, “Transmit Buffer Descriptor (TxBD).”

1 **Boldfaced** entries must be initialized by the user.

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 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 Section 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.

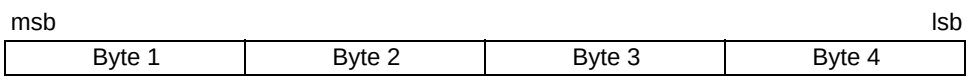


Figure 33-13. Mask1 Format

msb		lsb
Reserved, should be cleared.	Byte 5	

Figure 33-14. Mask2 Format

33.9.4.4.2 Comparison State Machine

The state machine shown in Figure 33-15 exists for filtering.

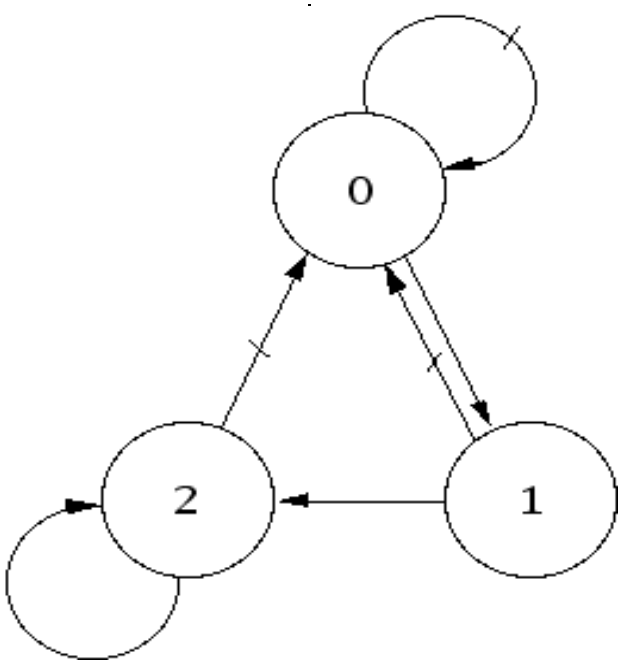


Figure 33-15. Filtering State Machine

- 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 e500 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.” 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[1]	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.

1 **Boldfaced** entries must be initialized by the user.

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.

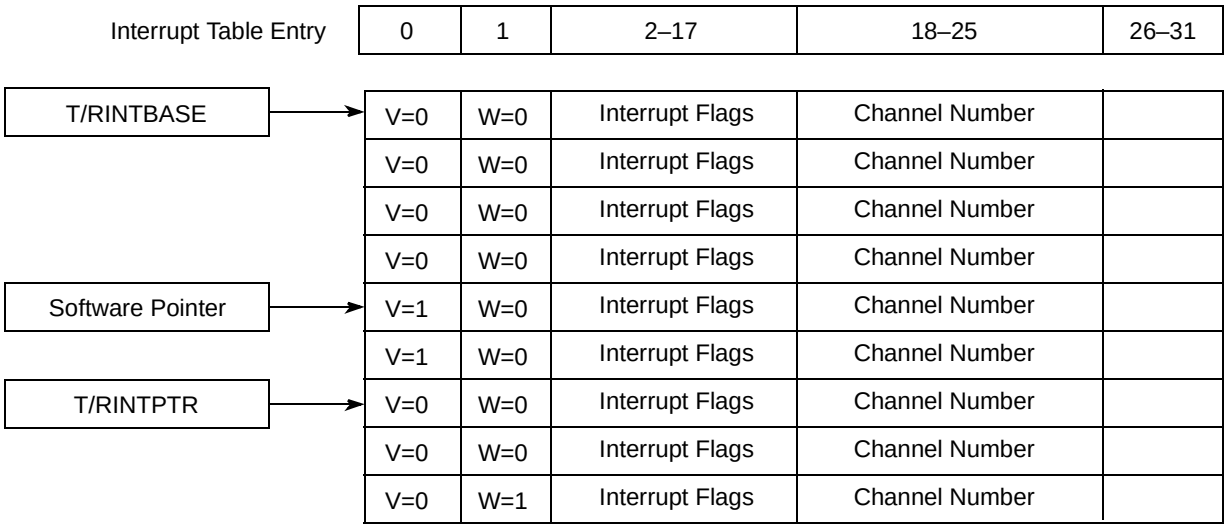


Figure 33-17. Interrupt Circular Table

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.

33.12.1 MCC 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.

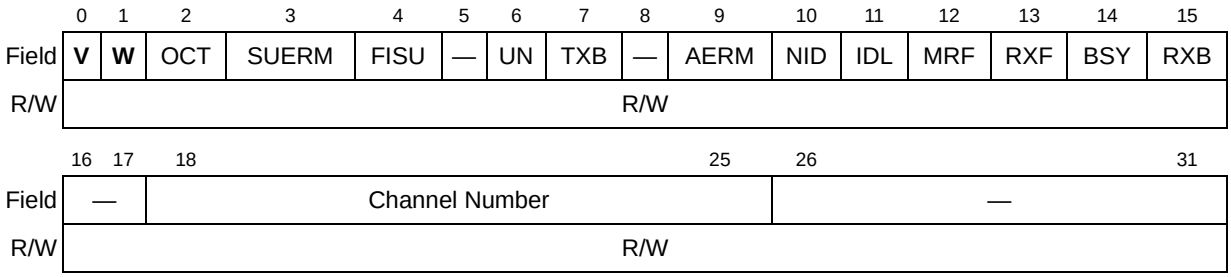


Figure 33-19. Interrupt Circular Table Entry

Table 33-18 describes interrupt circular table fields.

Table 33-18. Interrupt Circular Table Entry Field Descriptions

Bits	Name[1]	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.

Table 33-18. Interrupt Circular Table Entry Field Descriptions (continued)

Bits	Name[1]	Description
2	OCT	For SS7 only; otherwise reserved. N octets received. If the channel is in octet counting, this bit is set when N octets have been received.
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.

[1] **Boldfaced** entries must be initialized by the user.

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.

33.13.1 Receive Buffer Descriptor (RxBD)

Figure 33-20 shows the RxBD.

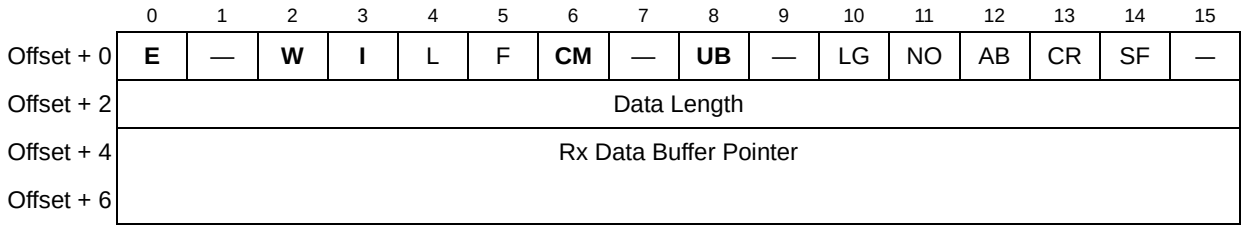


Figure 33-20. MCC Receive Buffer Descriptor (RxBD)

RxBD fields are described in Table 33-19.

Table 33-19. RxBD Field Descriptions

Bits	Name[1]	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).

Table 33-19. RxBD Field Descriptions (continued)

Bits	Name[1]	Description
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.
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.

Table 33-19. RxBD Field Descriptions (continued)

Bits	Name[1]	Description
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 octets.
15	—	Reserved, should be cleared.

1 **Boldfaced** entries must be initialized by the user.

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 $8 \times$ n if ECHAMR[TS]=0 and equal to $8 \times$ 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.

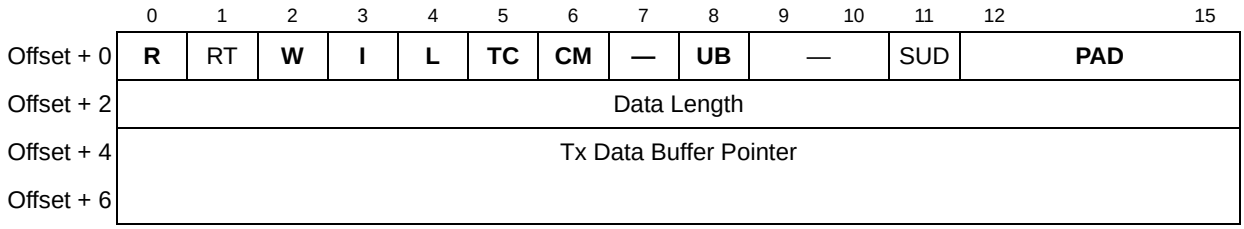


Figure 33-21. MCC Transmit Buffer Descriptor (TxBD)

Table 33-20 describes TxBD fields.

Table 33-20. TxBD Field Descriptions

Bits	Name[1]	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.
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.

Table 33-20. TxBD Field Descriptions (continued)

Bits	Name[1]	Description
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.

1 **Boldfaced** entries must be initialized by the user.

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," 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.

NOTE

This clock ratio is based on the hardware architecture and does not ensure that an application will run at that speed. It is the responsibility of the system designer to check AC specifications of the I/O pins and determine the maximum frequency.

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.

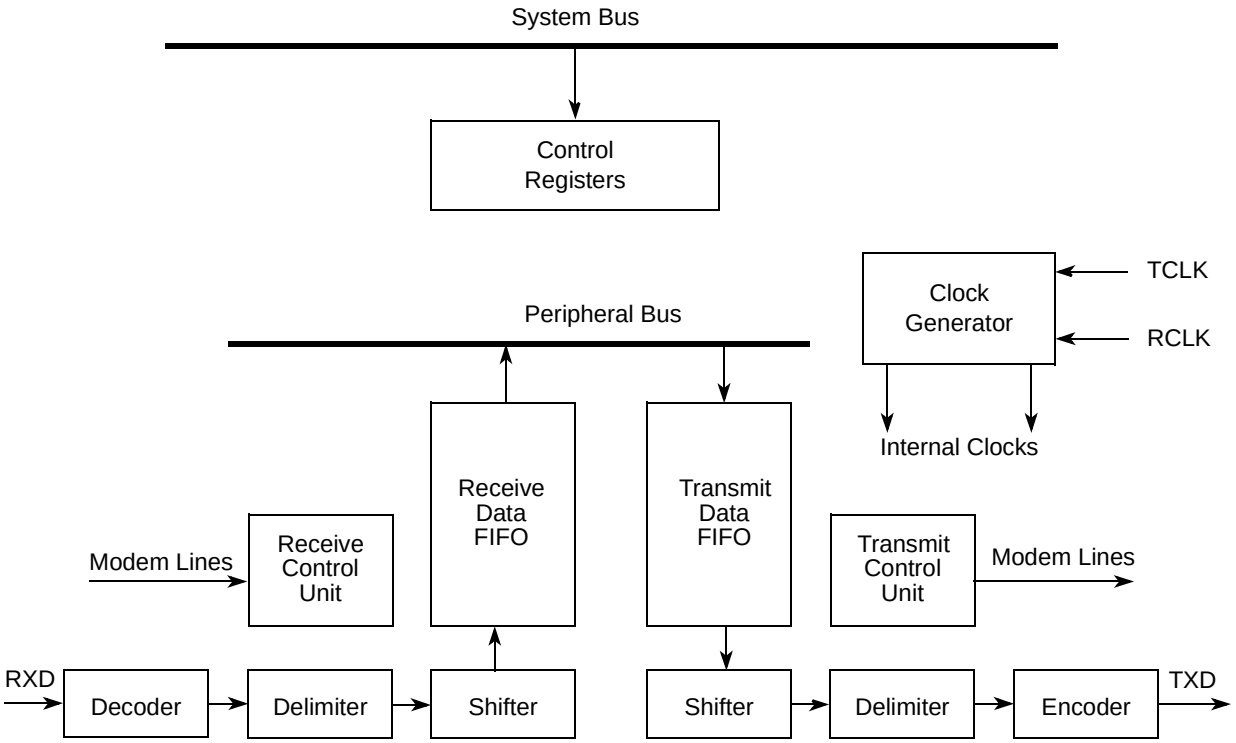


Figure 34-1. FCC Block Diagram

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 45, "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 Ethernet 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	$\overline{\text{CD}}$ pulse (transparent mode only) 0 Normal operation (envelope mode). $\overline{\text{CD}}$ should envelope the frame; to negate $\overline{\text{CD}}$ while receiving causes a $\overline{\text{CD}}$ lost error. 1 Pulse mode. Once $\overline{\text{CD}}$ is asserted (high to low transition), synchronization has been achieved, and further transitions of $\overline{\text{CD}}$ do not affect reception. CDP must be set if this FCC is used with the TSA in transparent mode.
6	CTSP	$\overline{\text{CTS}}$ pulse 0 Normal operation (envelope mode). $\overline{\text{CTS}}$ should envelope the frame; to negate $\overline{\text{CTS}}$ while transmitting causes a $\overline{\text{CTS}}$ lost error. See Section 34.12, “FCC Timing Control.” 1 Pulse mode. $\overline{\text{CTS}}$ is asserted when synchronization is achieved; further transitions of $\overline{\text{CTS}}$ do not affect transmission. When running HDLC, the FCC samples $\overline{\text{CTS}}$ only once before sending the first frame after the transmitter is enabled (ENT = 1).
7	CDS	$\overline{\text{CD}}$ sampling 0 The $\overline{\text{CD}}$ 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 $\overline{\text{CD}}$ input is assumed to be synchronous with the data, giving faster operation. In this mode, $\overline{\text{CD}}$ must transition while the receive clock is in the low state. When $\overline{\text{CD}}$ goes low, data is received. This is useful when connecting MPC8560s in transparent mode since it allows the $\overline{\text{RTS}}$ signal of one MPC8560 to be connected directly to the $\overline{\text{CD}}$ signal of another MPC8560.
8	CTSS	$\overline{\text{CTS}}$ sampling 0 The $\overline{\text{CTS}}$ 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 $\overline{\text{CTS}}$ input is assumed to be synchronous with the data, giving faster operation. In this mode, $\overline{\text{CTS}}$ must transition while the transmit clock is in the low state. As soon as $\overline{\text{CTS}}$ is low, data transmission begins. This mode is useful when connecting MPC8560 in transparent mode because it allows the $\overline{\text{RTS}}$ signal of one MPC8560 to be connected directly to the $\overline{\text{CTS}}$ signal of another MPC8560.
9–15	—	Reserved, should be 0.

Table 34-1. GFMR Register Field Descriptions (continued)

Bits	Name	Description
16–17	SYNL	Sync length (transparent mode only). Determines the operation of an FCC receiver configured for totally transparent operation only. See Section 42.3.1, "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).
18	RTSM	$\overline{RTS}$ mode 0 Send idles between frames as defined by the protocol. $\overline{RTS}$ is negated between frames (default). 1 Send flags/syncs between frames according to the protocol. $\overline{RTS}$ 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	0111_1110_0111_1110	
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.

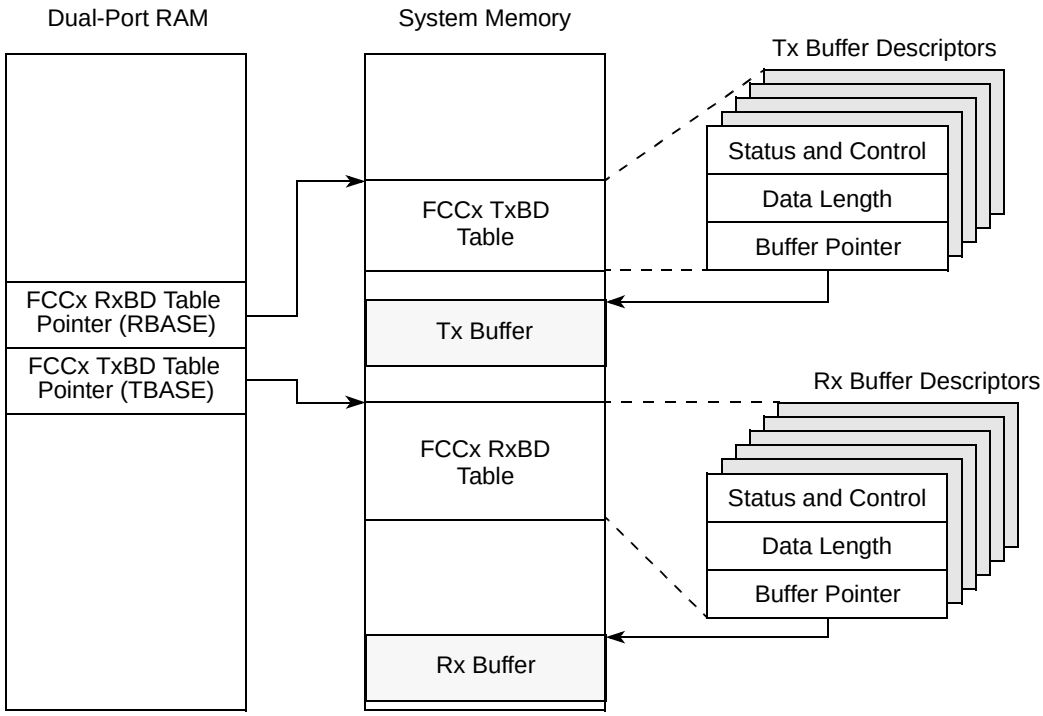


Figure 34-5. FCC Memory Structure

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.

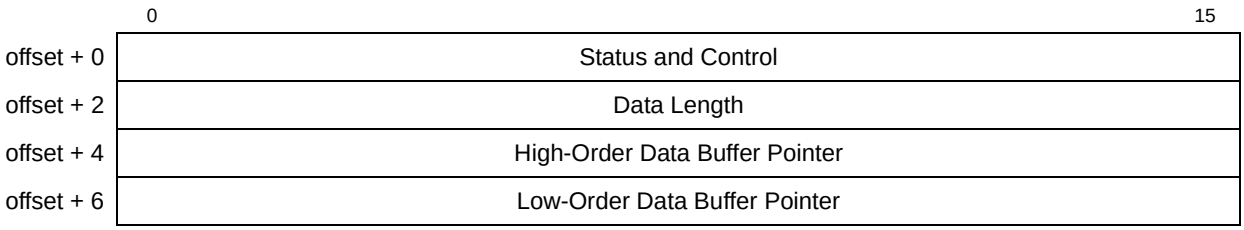


Figure 34-6. Buffer Descriptor Format

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[2]	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.

Table 34-5. FCC Parameter RAM Common to All Protocols except ATM (continued)

Offset[1]	Name[2]	Width	Description
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.
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."
2 **Boldfaced** entries must be initialized by the user.

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 [1]	Description
0	—	Reserved, should be cleared.
1	**FCCP**	FCC priority. 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.

[1] **Boldfaced** entries must be initialized by the user.

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 CPM interrupt controller; 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 45, “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.

34.11.1 FCC Transmit Errors

There are four errors in the FCC transmitter that make it necessary for software to act on the transmitter before correct operation can continue. The errors are as follows:

- CTS-lost indication in HDLC transmitter (use re-initialization procedure)

- Underrun in any of the serial FCC transmitter protocols (use re-initialization procedure)
- Late collision in fast Ethernet transmitter (use recovery or re-initialization procedure)
- Expiration of retry limit in fast Ethernet (use recovery or re-initialization procedure)

In addition to status bits in the current TxBD, these errors are reported via the TXE event bit in the FCCE as a convenience for the user to implement error handling. TXE is a safe indication that a recovery or re-initialization procedure must be started.

34.11.1.1 Re-Initialization Procedure

1. Disable the FCC transmission by clearing GFMR[ENT].
2. Remember the TBPTR value taken from the FCC parameter RAM.
3. Issue an “INIT TX PARAMS” command using the CPCR.
4. Restore the remembered TBPTR into the FCC parameter RAM.
5. Adjust TxBD handling as described in Section 34.11.1.3, “Adjusting Transmitter BD Handling.”
6. Enable FCC transmission by setting GFMR[ENT].

34.11.1.2 Recovery Sequence

1. Adjust TxBD handling as described in Section 34.11.1.3, “Adjusting Transmitter BD Handling.”
2. Issue a “RESTART TX” command using the CPCR.

34.11.1.3 Adjusting Transmitter BD Handling

When a TXE event occurs, the TBPTR may already point beyond BDs still marked as ready due to internal pipelining. If the TBPTR is not adjusted, these BDs would be skipped while still being marked as ready. Software must determine if these BDs should be retransmitted or if they should be skipped, depending on the protocol and application needs. This requires the following steps:

1. From the current TBPTR value, search backwards over all (if any) BDs still marked as ready to find the first BD that has not been closed by the CPM. The search process should stop if the BD to be checked next is not ready or if it is the most recent BD marked as ready by the CPU transmit software. This is to avoid an endless loop in case the CPU software fills the BD ring completely.
2. A) For skipping BDs, manually close all BDs from the BD just found up to and including the BD just before TBPTR. Leave the TBPTR value untouched.

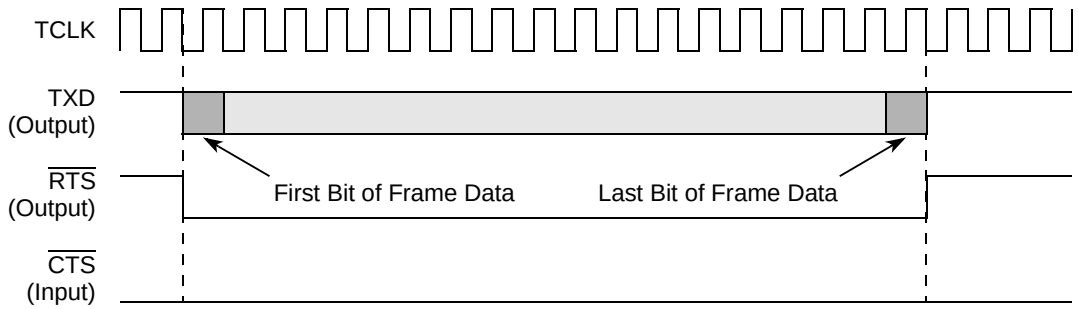
   B) For retransmitting BDs, change the TBPTR value to point to the BD just found.

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].

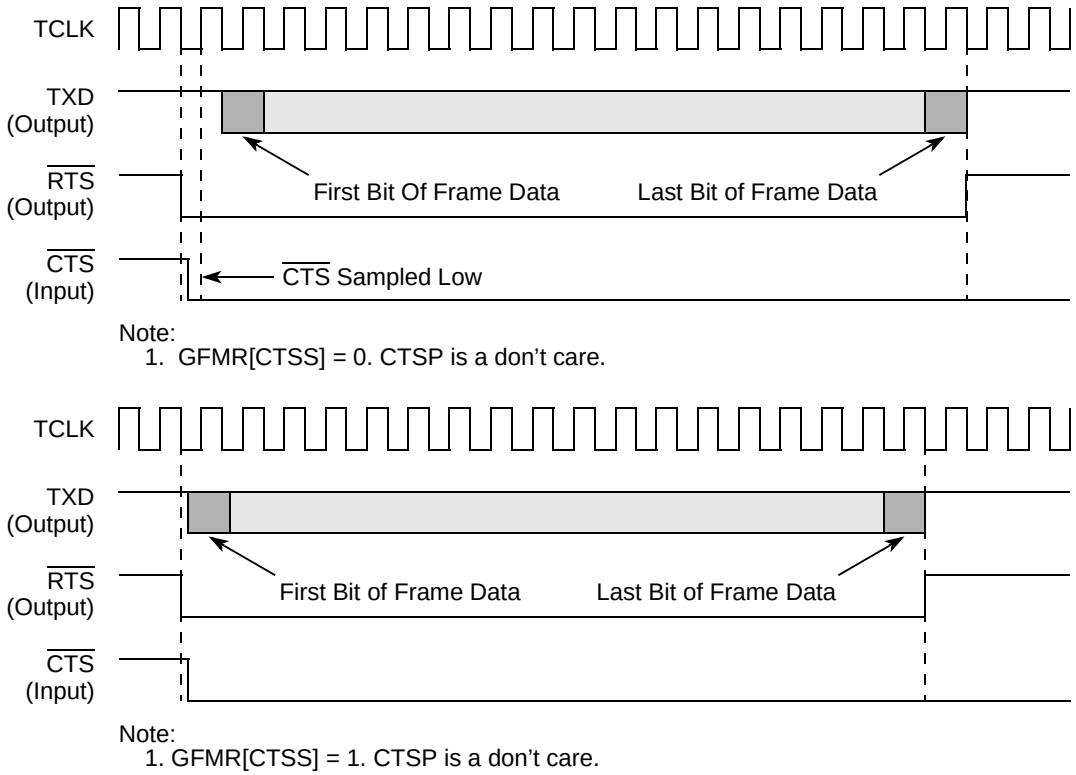


Figure 34-9. Output Delay from $\overline{\text{CTS}}$ Asserted

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.

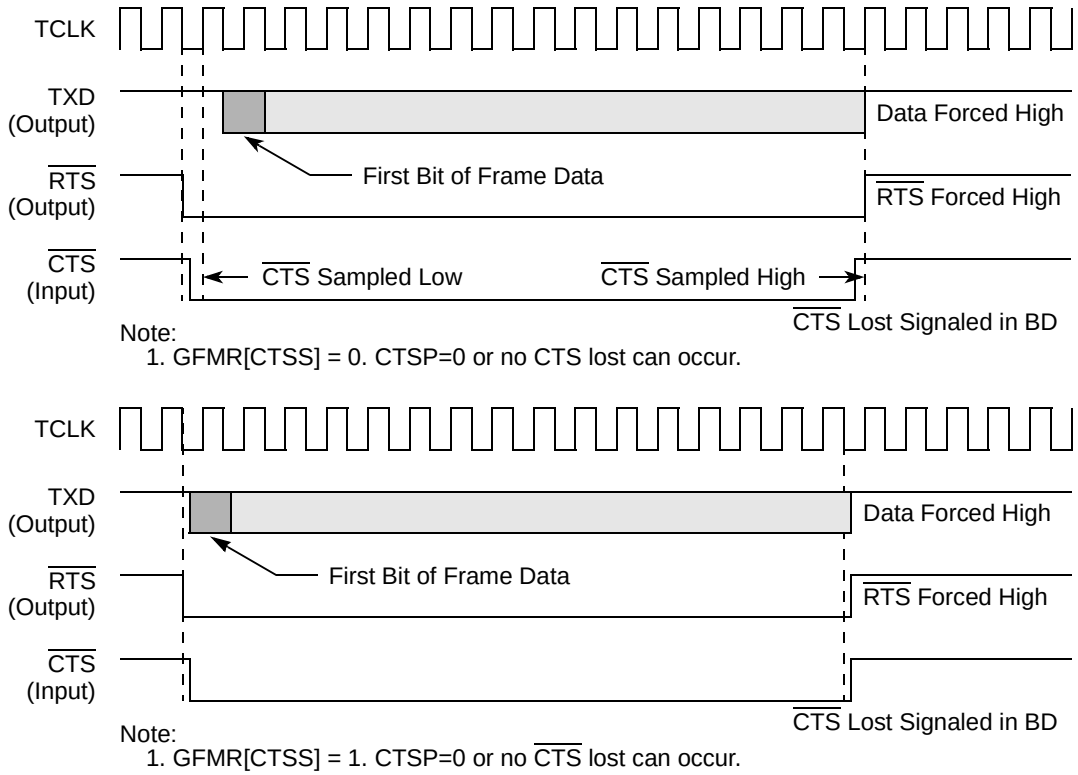


Figure 34-10. $\overline{\text{CTS}}$ Lost

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.

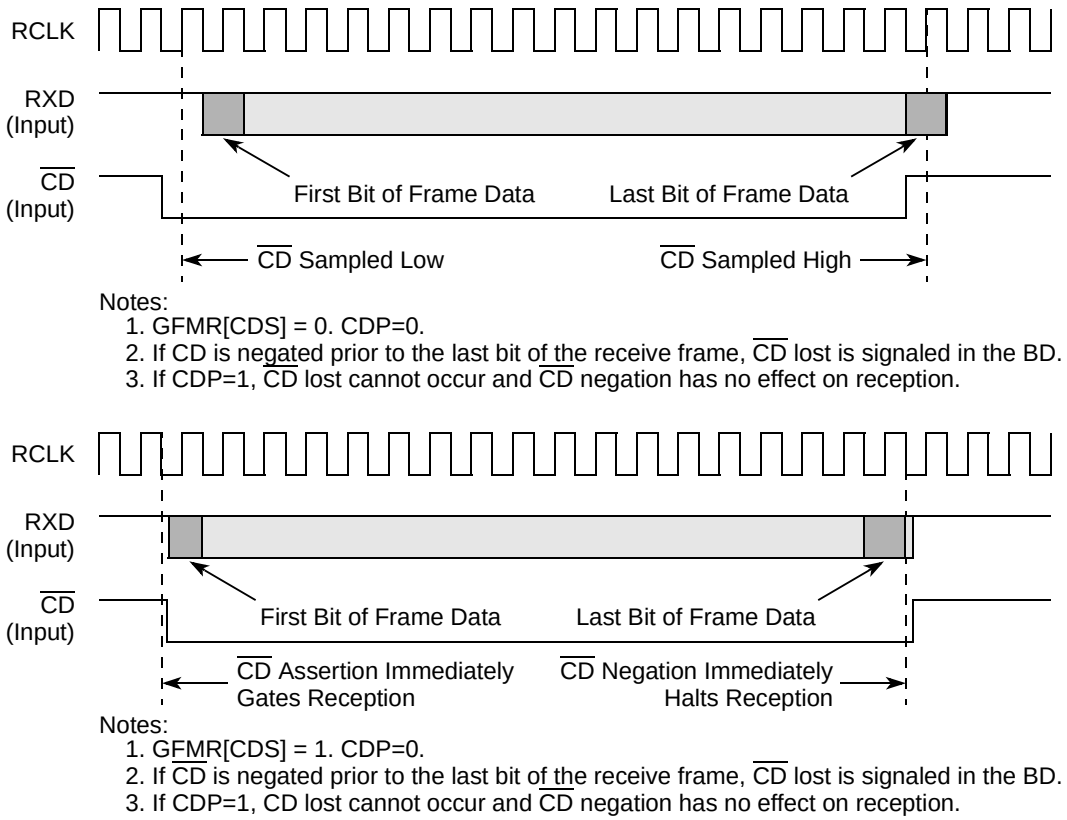


Figure 34-11. Using $\overline{\text{CD}}$ to Control Reception

If it is programmed to envelope data, $\overline{\text{CD}}$ must remain asserted during frame transmission or a $\overline{\text{CD}}$ lost error occurs. The negation of $\overline{\text{CD}}$ terminates reception. If [CDS] = 0, $\overline{\text{CD}}$ must be sampled by the FCC before a $\overline{\text{CD}}$ lost is recognized. Otherwise, the negation of $\overline{\text{CD}}$ immediately causes the $\overline{\text{CD}}$ lost condition.

Note that if GFMR[CDS] = 1, all $\overline{\text{CD}}$ 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.

34.13.1 FCC 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.

34.13.2 FCC 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].

34.13.3 FCC 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 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.

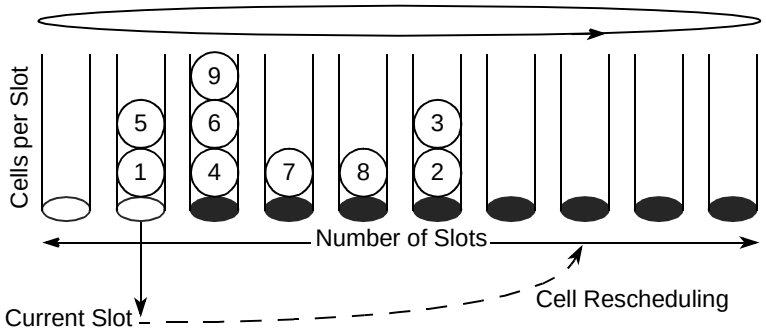


Figure 35-1. APC Scheduling Table Mechanism

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 = 155.52 Mbps/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 1216.

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

$$\text{Min bit rate} = \frac{\text{line rate}}{(\text{number\_of\_slots} - 1) \times \text{cells per slot}} \quad \text{(B)}$$

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:

Max bit rate = (155.52 Mbps)/8 = 19.44 Mbps

Min bit rate = (155.52 Mbps)/(1024 × 8) = 18.98 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.

$$\text{Rate [slots]} = \frac{\text{line rate [bps]}}{\text{VC rate [bps]} \times \text{cells per slot}} \quad \text{(C)}$$

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 the burst tolerance (BT), as shown in Figure 35-2. (The GCRA is also known as the leaky bucket algorithm.)

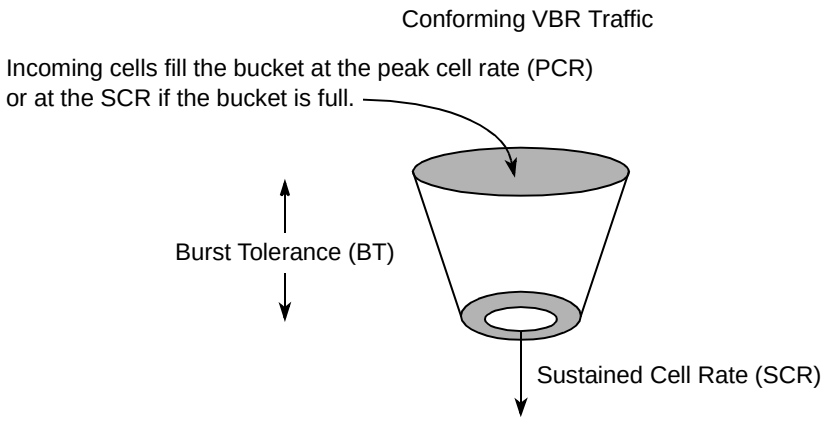


Figure 35-2. VBR Pacing Using the GCRA (Leaky Bucket Algorithm)

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].

$$\begin{aligned} BT\,[slots] &= (MBS[cells] - 2) \times (SCR[slots] - PCR[slots]) + SCR[slots] \\ &= (1000 - 2) \times ((9 + 185/256) - (3 + 62/256)) + (9 + 185/256) \\ &= 6477 \end{aligned} \qquad \text{(D)}$$

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.

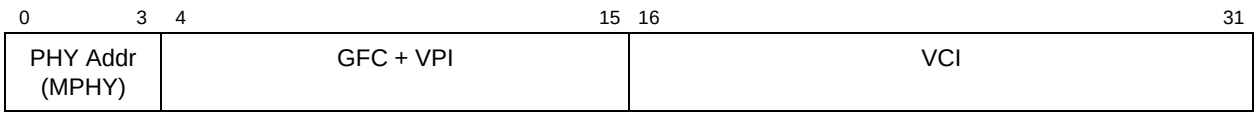


Figure 35-3. External CAM Data Input Fields

The output of the CAM, shown in Figure 35-4, is a 32-bit entry (16-bit channel code and a match-status bit).

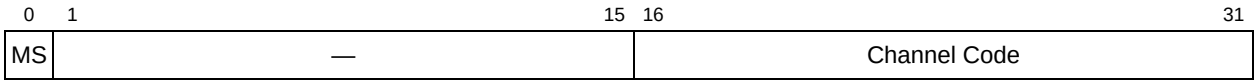


Figure 35-4. External CAM Output Fields

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, 2 CAM devices are needed if using more than 16 PHYs.The msb of the PHY address (bit 4) selects between the 2 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.

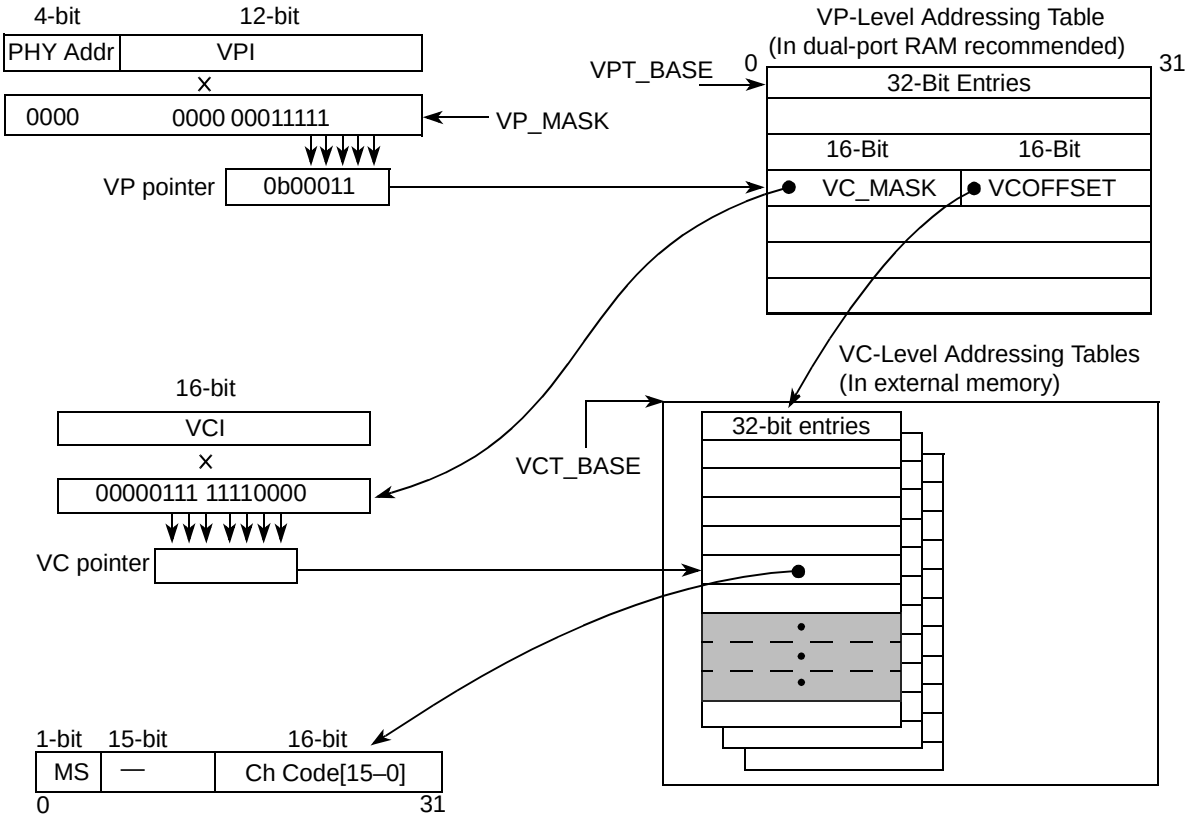


Figure 35-5. Address Compression Mechanism

Figure 35-5 shows VP_MASK selecting 5 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, 2 sets of look-up tables are needed if using more than 16 PHYs.The msb of the PHY address (bit 4) selects between the 2 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 (continued)

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 10 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: $VCOFFSET_{(n+1)} = VCOFFSET_n + 2^{(\text{number of ones in VC\_MASK}n)}$

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.

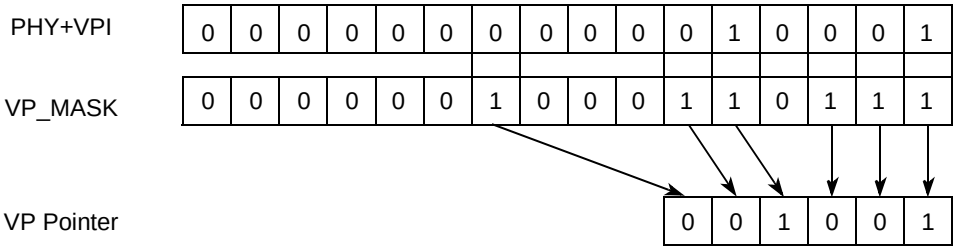


Figure 35-7. VP Pointer Address Compression

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.

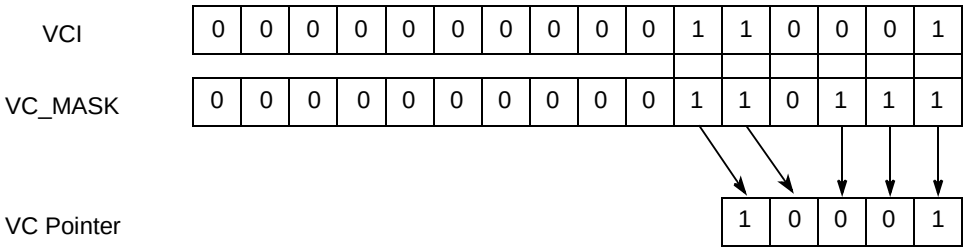


Figure 35-8. VC Pointer Address Compression

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.

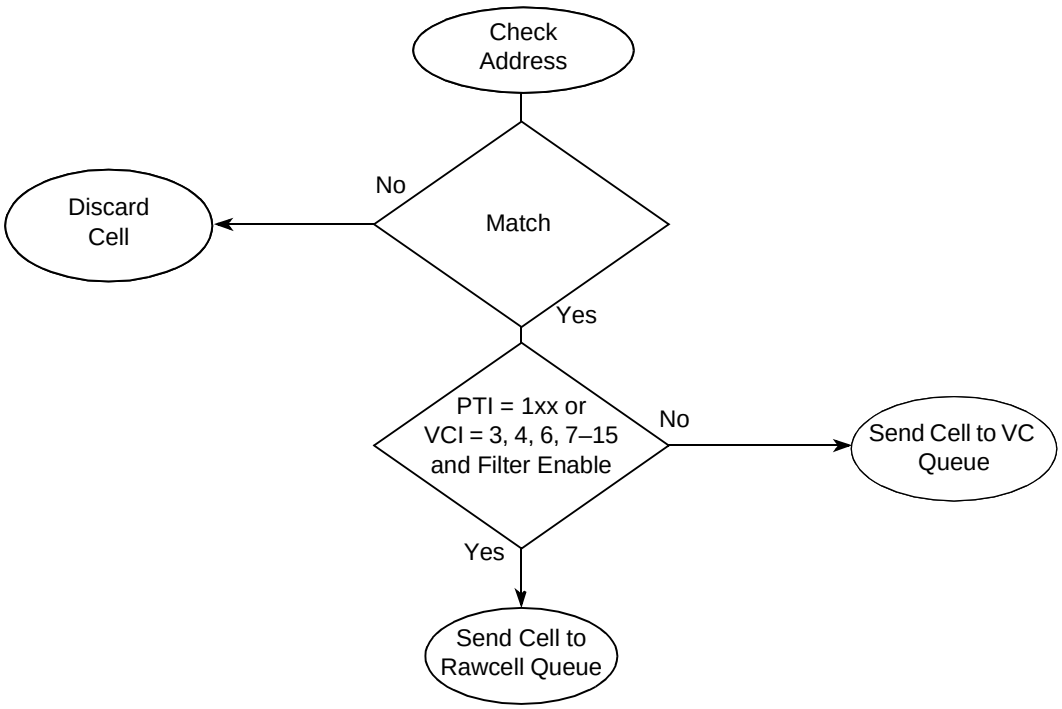


Figure 35-9. ATM Address Recognition Flowchart

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 ABR Model

Figure 35-10 shows the MPC8560 ABR model.

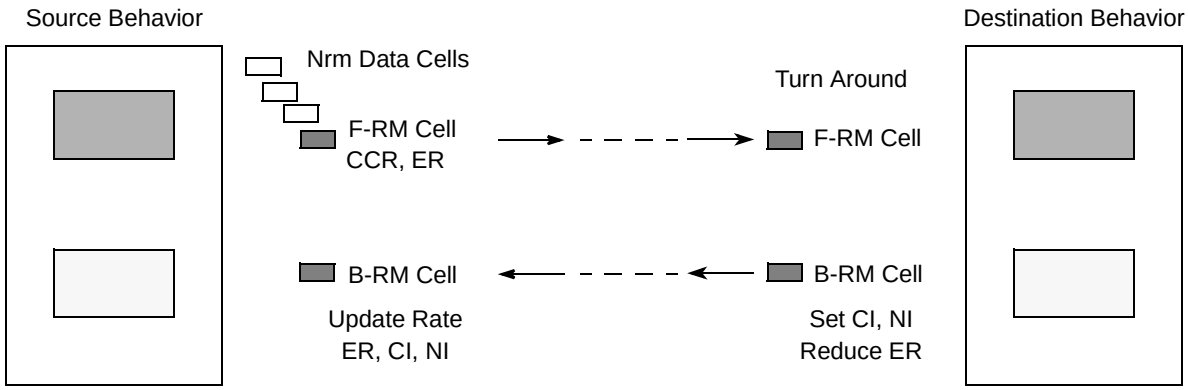


Figure 35-10. MPC8560 ABR Basic Model

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 cell.
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.

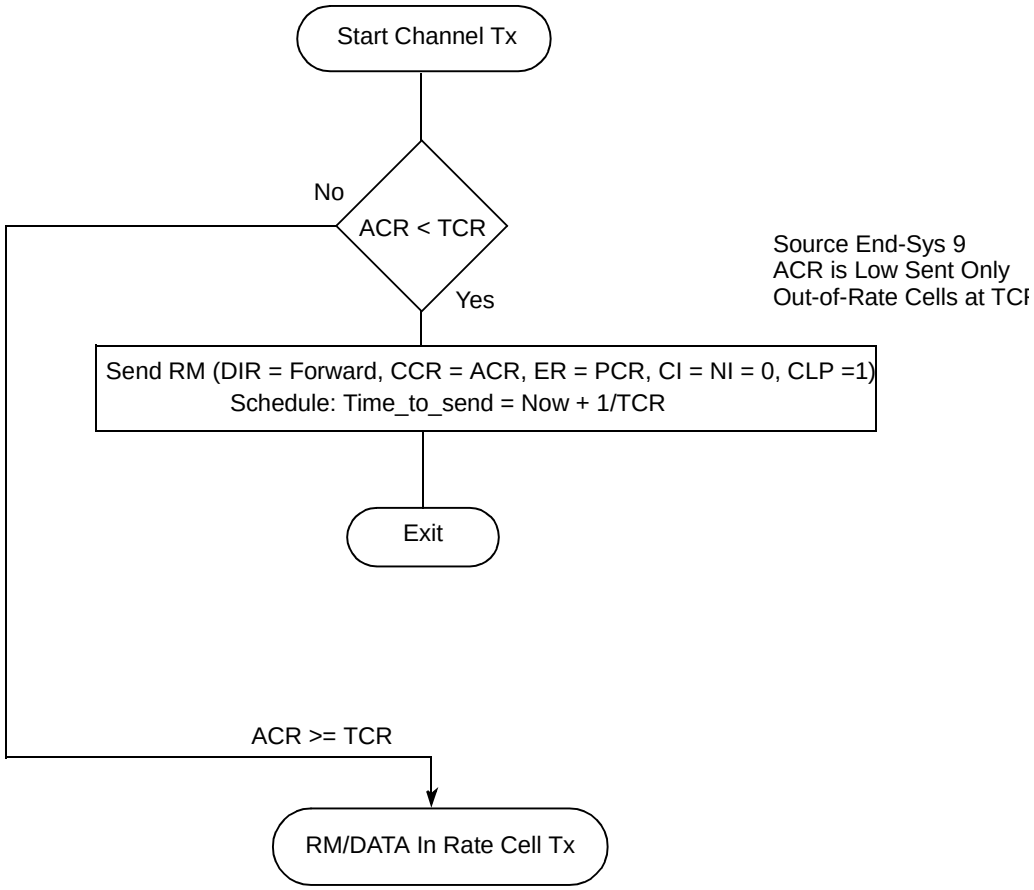


Figure 35-11. ABR Transmit Flow

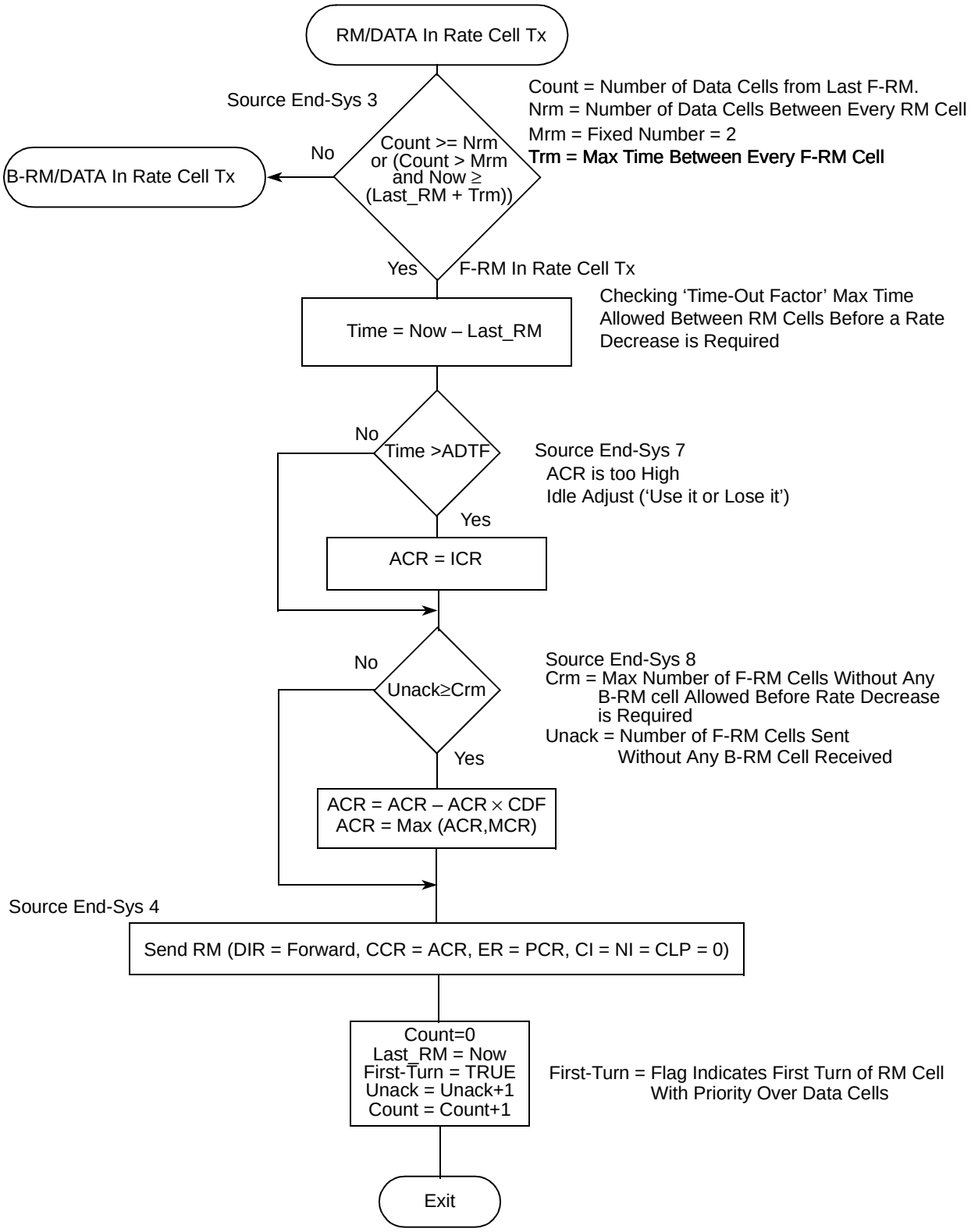


Figure 35-12. ABR Transmit Flow (continued)

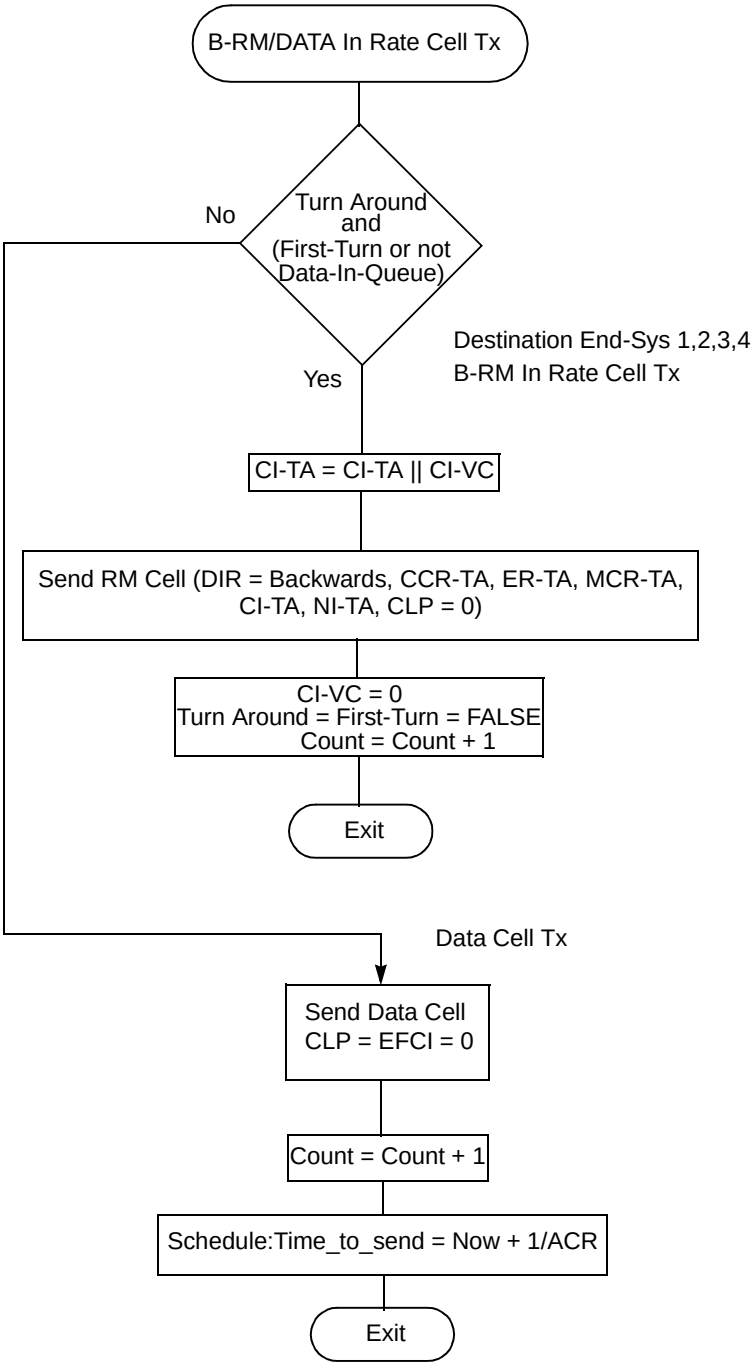


Figure 35-13. ABR Transmit Flow (continued)

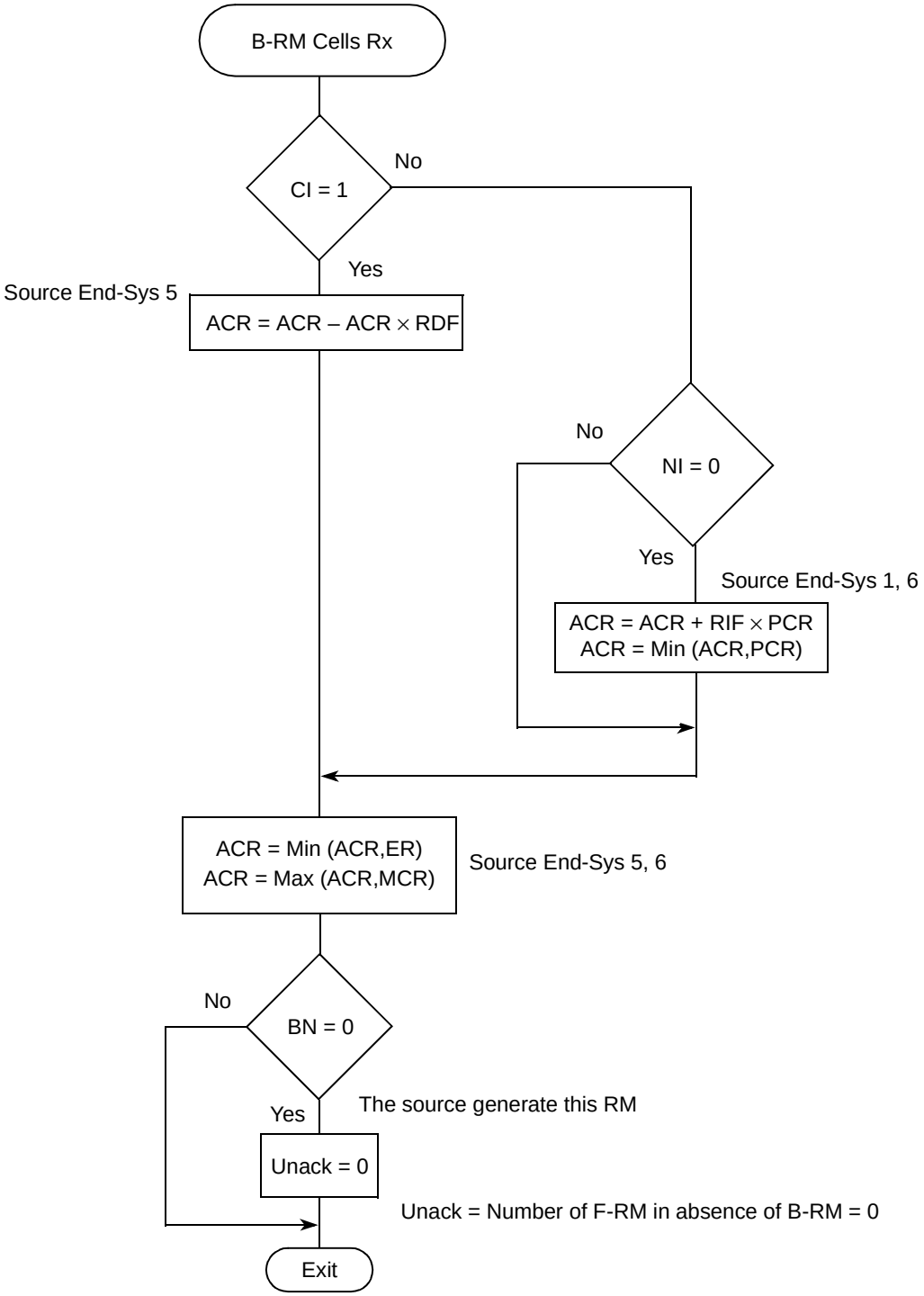


Figure 35-14. ABR Receive Flow

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.

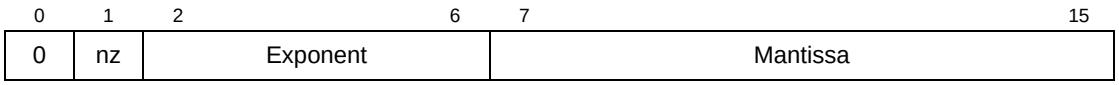


Figure 35-15. Rate Format for RM Cell

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.

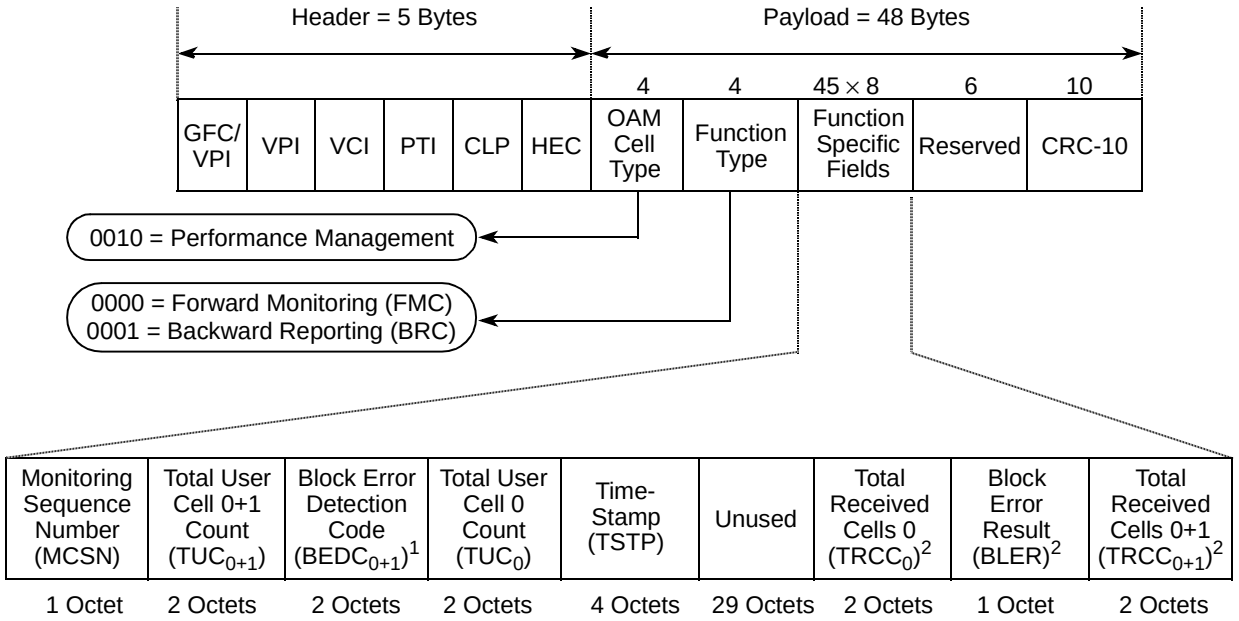
A series of OAM cells can be sent using one ATM TRANSMIT command by creating a table of AAL0 TxBDs. If the channel’s TCT[AVCF] (auto VC off) is set, the transmitter automatically removes it from the APC (that is, it does not generate periodic transmit requests for this channel after all AAL0 BDs are 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.



Notes:

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 1 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, “Running a Performance Block Test”), 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, and $BEDC_{0+1}$-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.

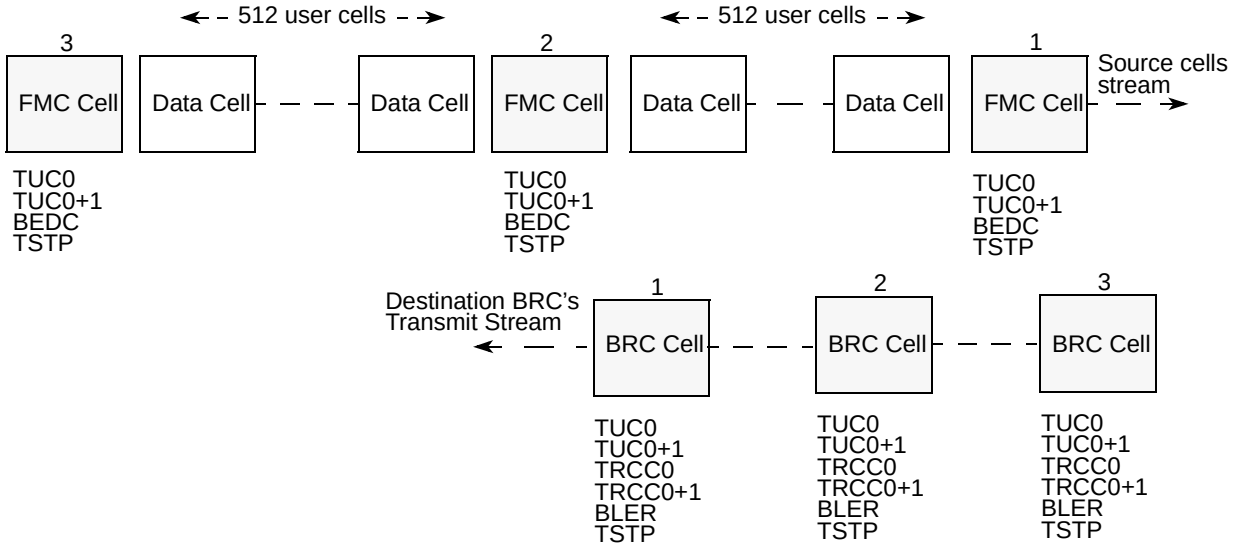


Figure 35-18. FMC, BRC Insertion

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.

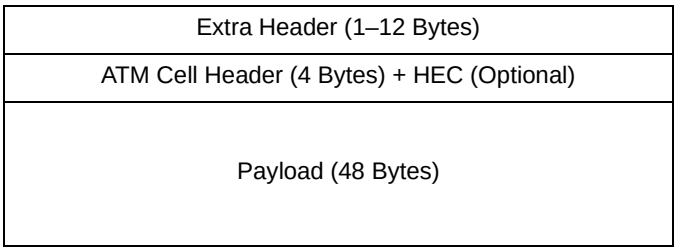


Figure 35-19. Format of User-Defined Cells

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, 2 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.”

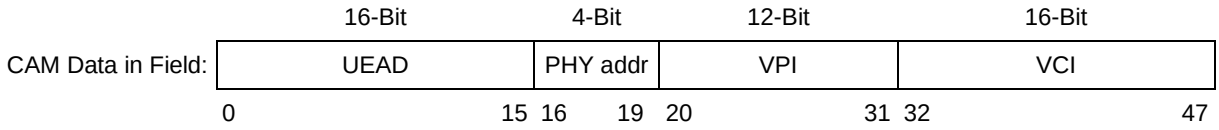


Figure 35-20. External CAM Address in UDC Extended Address Mode

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:

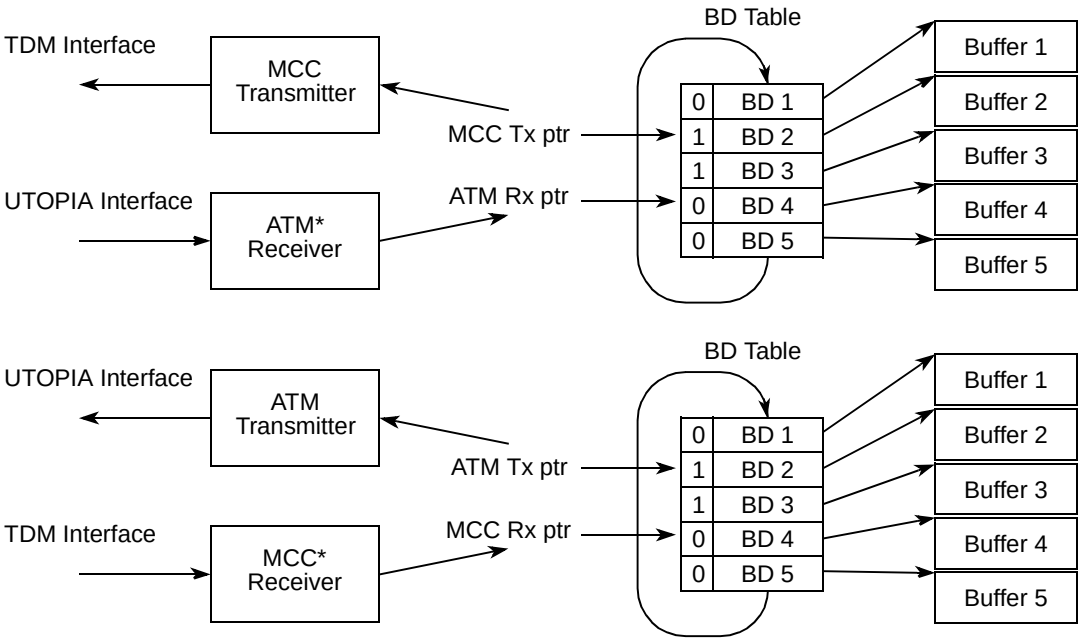
- Circuit emulation service (CES)
- Carrying voice over ATM
- Multiplexing several low speed services, such as voice and data, onto one ATM connection

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, 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 programmed 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 setup 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. Figure 35-22 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

Figure 35-22. 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 35-23.

	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 35-23. VCI Filtering Enable Bits

Table 35-12 describes the operation of the VCI filtering enable bits.

Table 35-12. 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 35-24 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 35-24. Global Mode Entry (GMODE)

Table 35-13 describes GMODE fields.

Table 35-13. 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.

Table 35-13. GMODE Field Descriptions (continued)

Bits	Name	Description
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.
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 setup. 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 35-14 lists sizes for RCT, TCT, and TCTE.

Table 35-14. 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 1, the second has channel code 2, and so on. Codes of 255 or less indicate internal channels; codes greater than 255 indicate external channels. Channel code 1 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 1024 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) = 4K). Channels 0–1 are reserved. The remaining 962 (1024 – 62) external channels are assigned channel codes 256–1217. See Figure 35-25.

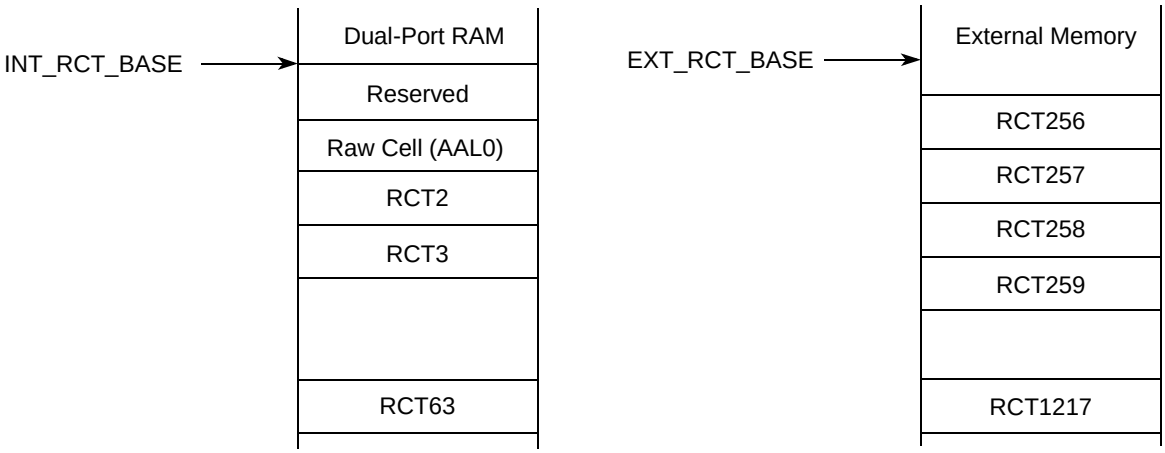


Figure 35-25. Example of a 1024-Entry Receive Connection Table

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 35-26 shows the format of an RCT entry.

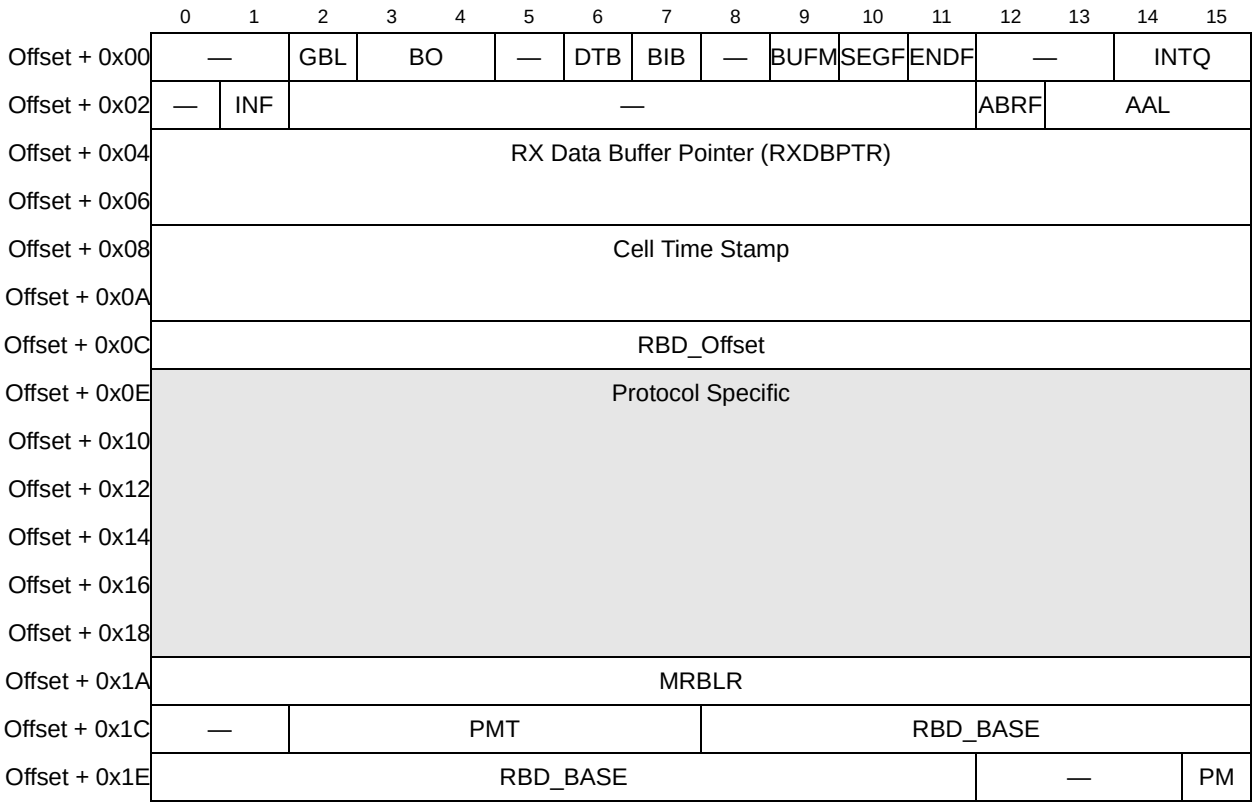


Figure 35-26. Receive Connection Table (RCT) Entry

Table 35-15 describes RCT fields.

Table 35-15. 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 system bus and the BD on the local bus, the SDMA accesses the UDC header on the system 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

Table 35-15. RCT Field Descriptions (continued)

Offset	Bits	Name	Description
	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
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 eight 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 35-27 shows the AAL5 protocol-specific area of an RCT entry.

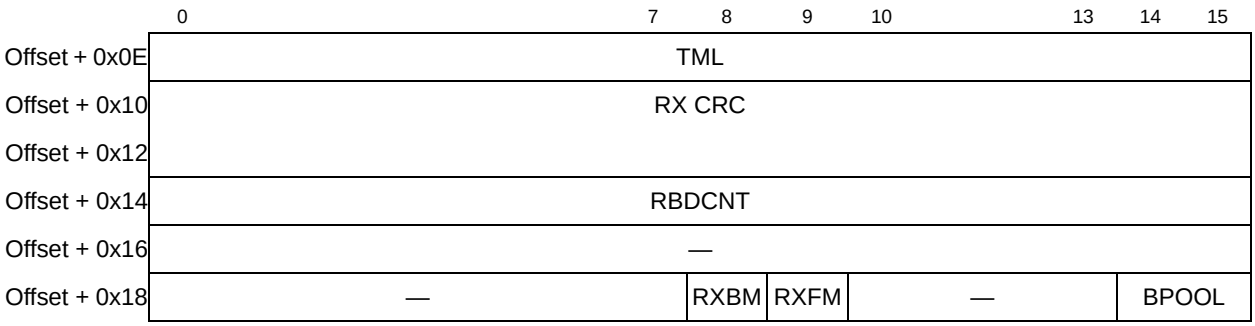


Figure 35-27. AAL5 Protocol-Specific RCT

Table 35-16 describes AAL5 protocol specific RCT fields.

Table 35-16. 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 35-28 shows the AAL5-ABR protocol-specific area of an RCT entry.

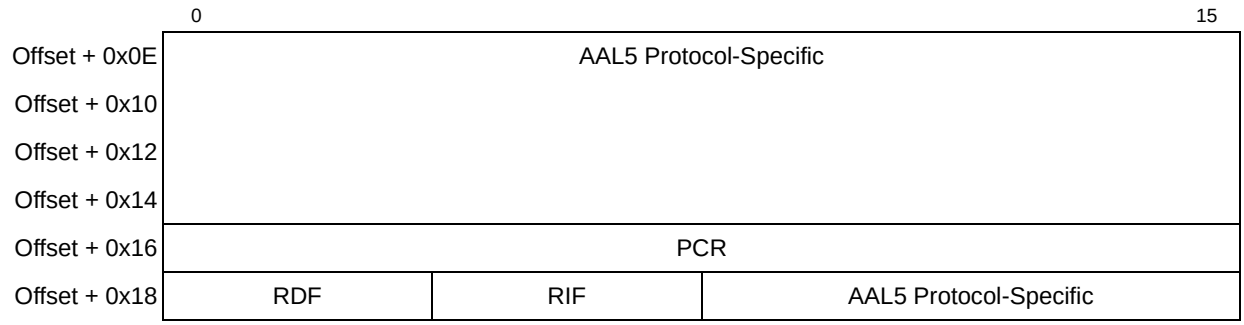


Figure 35-28. AAL5-ABR Protocol-Specific RCT

Table 35-17 describes AAL5-ABR protocol-specific RCT fields.

Table 35-17. 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.
0x18	0–3	RDF	Rate decrease factor for the current ABR channel. Controls the decrease in cell transmission rate on 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 35-29 shows the AAL1 protocol-specific area of an RCT entry.

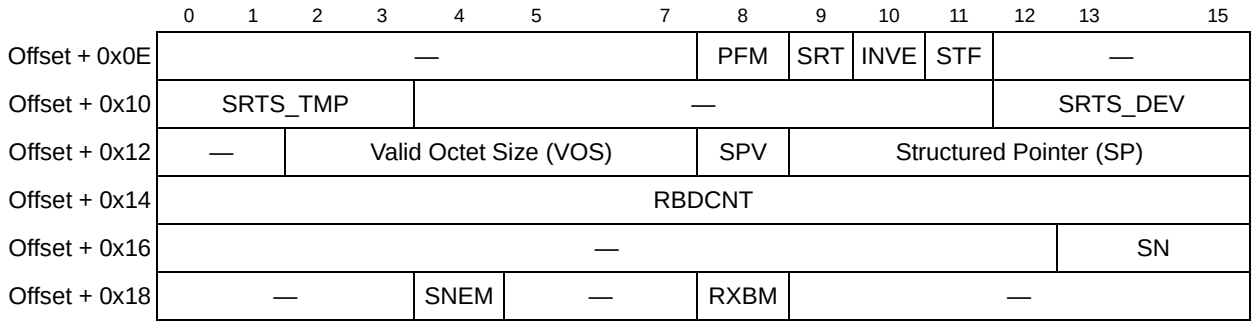


Figure 35-29. AAL1 Protocol-Specific RCT

Table 35-18 describes AAL1 protocol-specific RCT fields.

Table 35-18. 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.

Table 35-18. AAL1 Protocol-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
	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 35-18. 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 35-30 shows the layout for the AAL0 protocol-specific RCT.

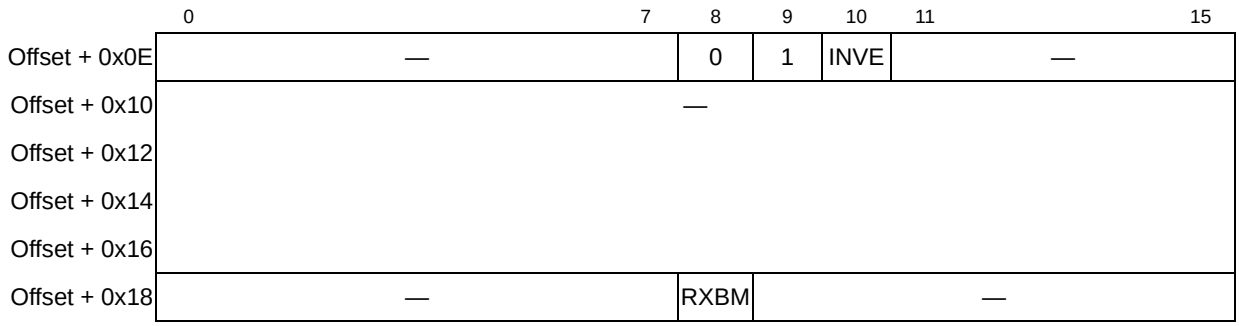


Figure 35-30. AAL0 Protocol-Specific RCT

Table 35-19 describes AAL0 protocol specific RCT fields.

Table 35-19. 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 35-19. 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 35-31 shows the format of an TCT entry.

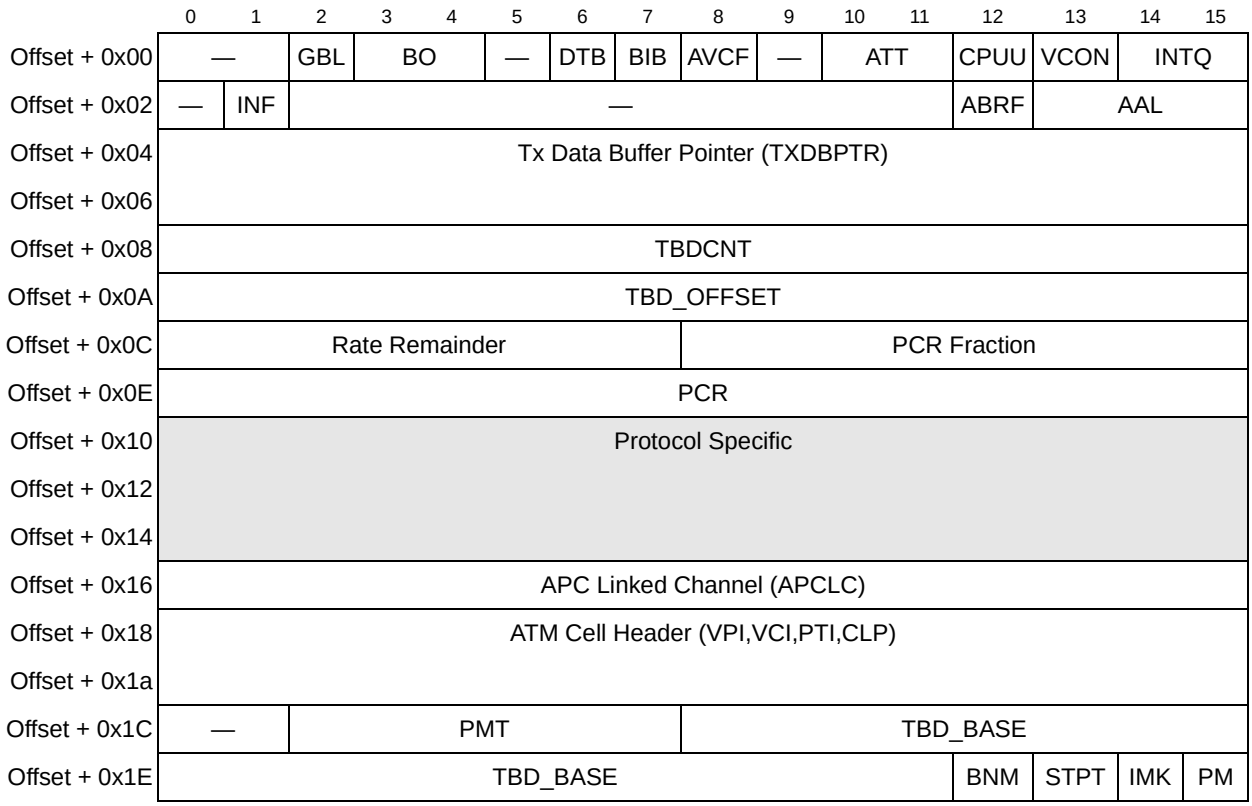


Figure 35-31. Transmit Connection Table (TCT) Entry

Table 35-20 describes general TCT fields.

Table 35-20. 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).

Table 35-20. TCT Field Descriptions (continued)

Offset	Bits	Name	Description
	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
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 35-20. 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 35-32 shows the AAL5 protocol-specific TCT.

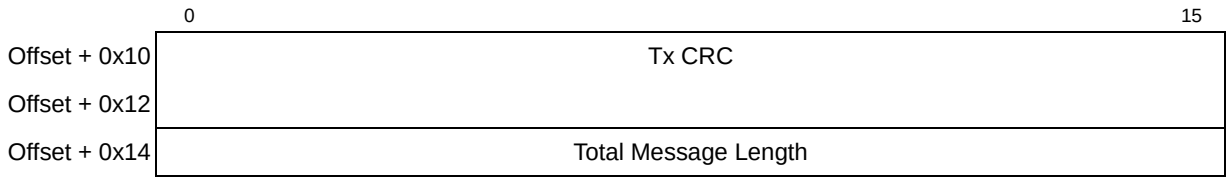


Figure 35-32. AAL5 Protocol-Specific TCT

Table 35-21 describes AAL5 protocol-specific TCT fields.

Table 35-21. 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 35-33 shows the AAL1 protocol-specific TCT.

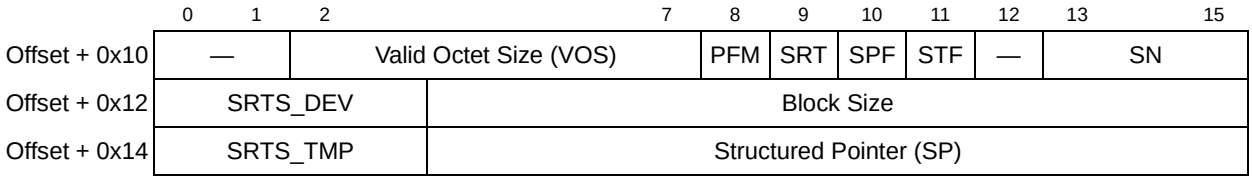


Figure 35-33. AAL1 Protocol-Specific TCT

Table 35-22 describes AAL1 protocol-specific TCT fields.

Table 35-22. 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 35-22. AAL1-Specific TCT Field Descriptions (continued)

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 4 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 35-34 shows the AAL0 protocol-specific TCT.

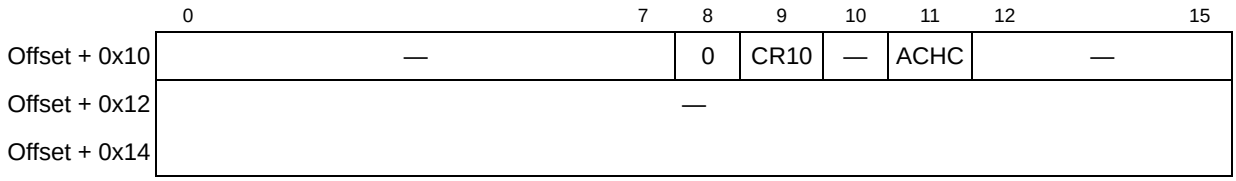


Figure 35-34. AAL0 Protocol-Specific TCT

Table 35-23 describes AAL0 protocol-specific TCT fields.

Table 35-23. 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 35-35 shows the VBR protocol-specific TCTE.

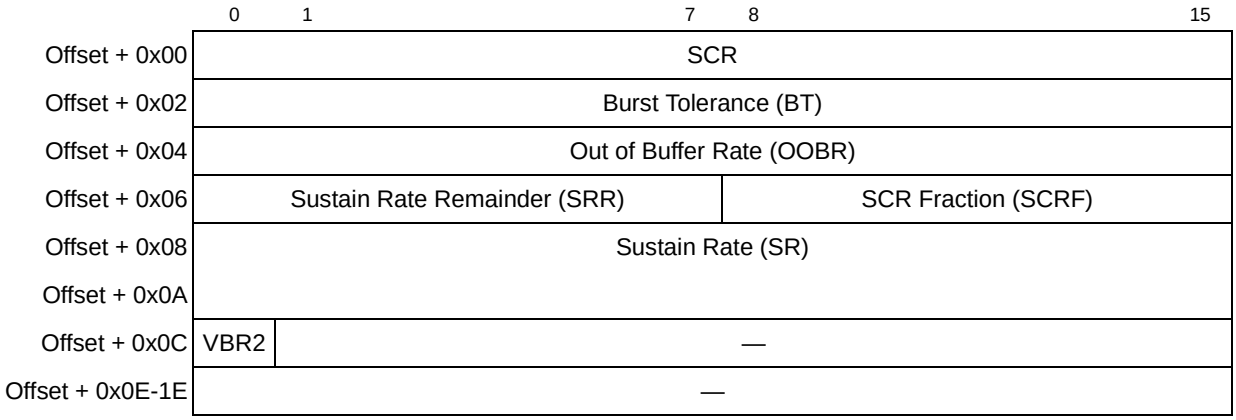


Figure 35-35. Transmit Connection Table Extension (TCTE)—VBR Protocol-Specific

Table 35-24 describes VBR protocol-specific TCTE fields.

Table 35-24. 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 35-36 shows the UBR+ protocol-specific TCTE.

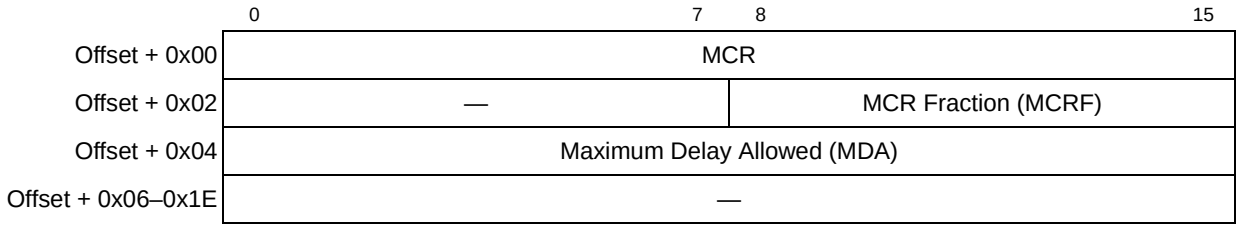


Figure 35-36. UBR+ Protocol-Specific TCTE

Table 35-25 describes UBR+ protocol-specific TCTE fields.

Table 35-25. 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 35-37 shows the ABR protocol-specific TCTE.

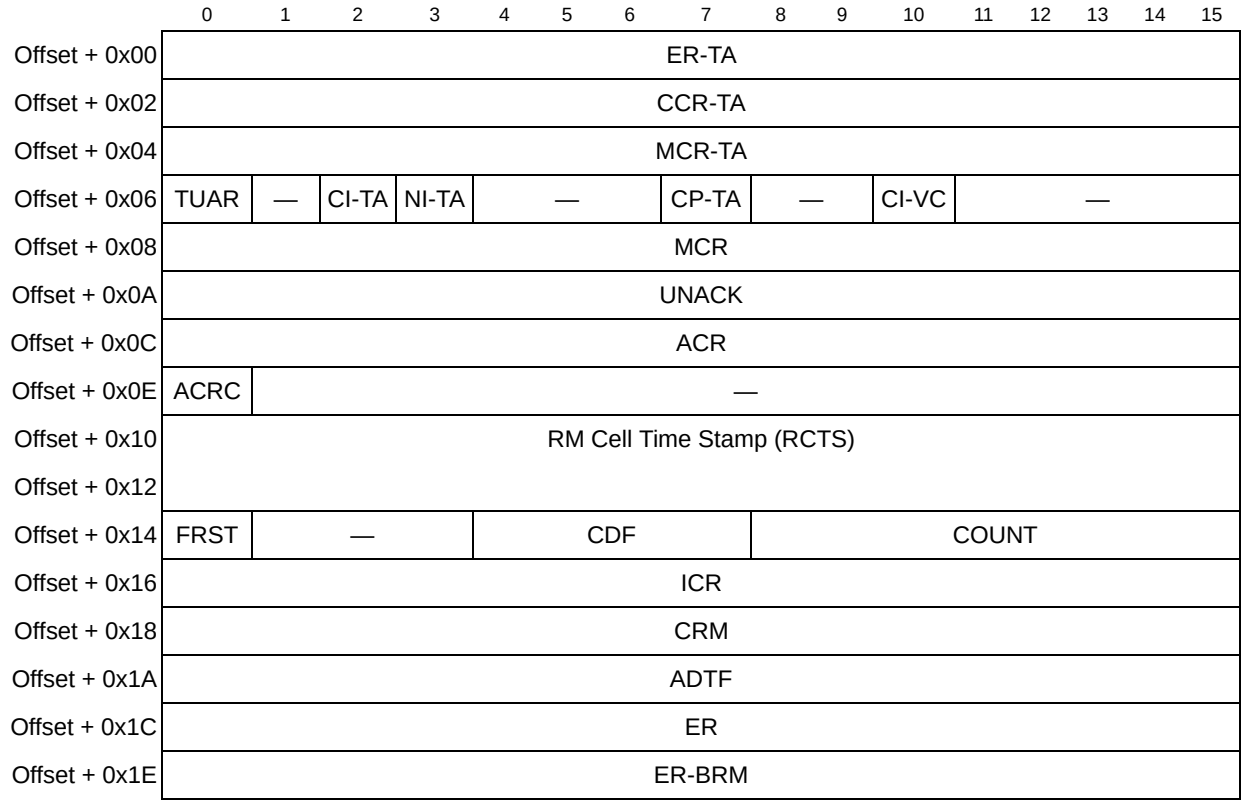


Figure 35-37. ABR Protocol-Specific TCTE

Table 35-26 describes ABR-specific TCTE fields.

Table 35-26. 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.

Table 35-26. ABR-Specific TCTE Field Descriptions (continued)

Offset	Bits	Name	Description
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.
	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.

Table 35-26. ABR-Specific TCTE Field Descriptions (continued)

Offset	Bits	Name	Description
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.
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 35-38. 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.

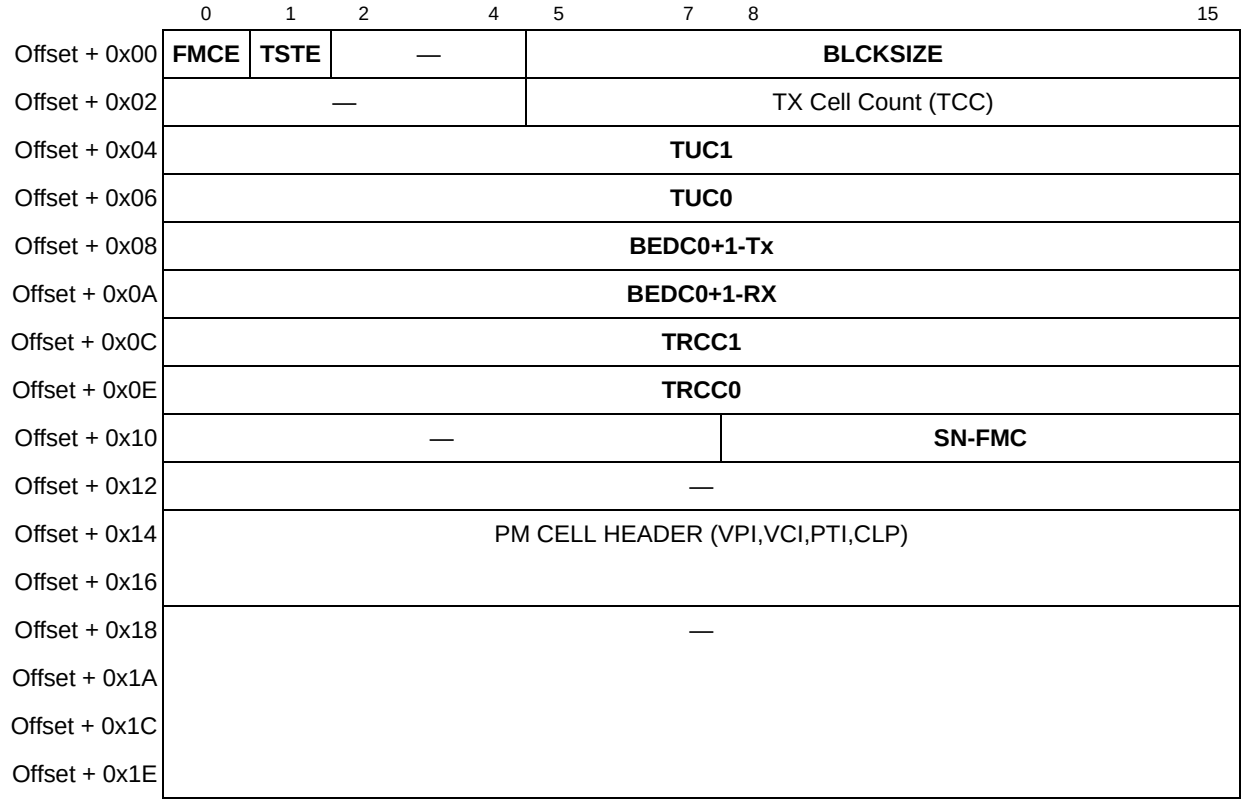


Figure 35-38. OAM Performance Monitoring Table

Table 35-27 describes fields in the performance monitoring table.

Table 35-27. OAM—Performance Monitoring Table Field Descriptions

Offset	Bits	Name[1]	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 1s. 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.

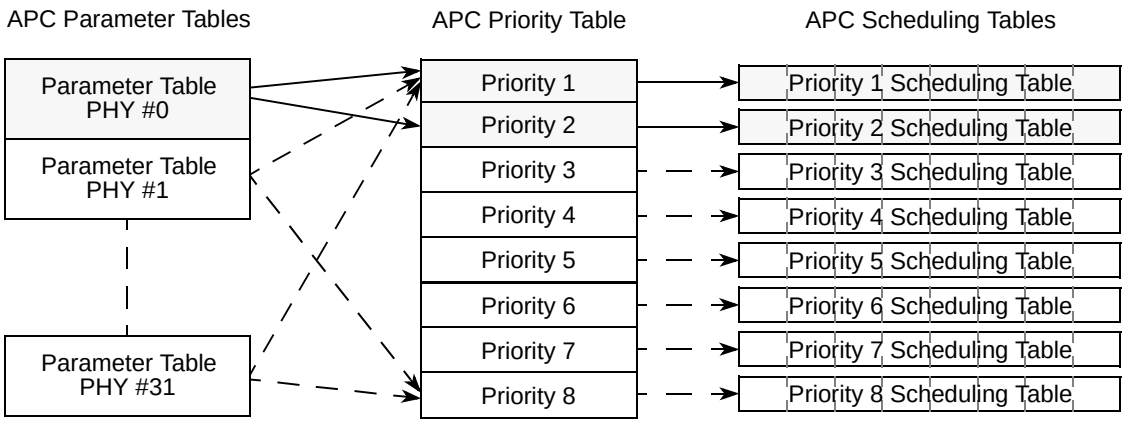
Table 35-27. OAM—Performance Monitoring Table Field Descriptions (continued)

Offset	Bits	Name[1]	Description
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

1 **Boldfaced** entries must be initialized by the user.

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 35-39.



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 35-39. ATM Pace Control Data Structure

35.10.4.1 APC Parameter Tables

Each PHY's APC parameter table, shown in Table 35-28, 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# × 32. Note however that in slave mode with multiple PHYs, the parameter table always resides at APCP_BASE regardless of the PHY address.

Table 35-28. APC Parameter Table

Offset[1]	Name[2]	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 × (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.)
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.

2 **Boldfaced** entries must be initialized by the user.

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 35-29 shows the structure of a priority table entry.

Table 35-29. 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 35-40. Each slot is a half-word entry. Note that the APC scheduling tables should be cleared before the APC unit is enabled.

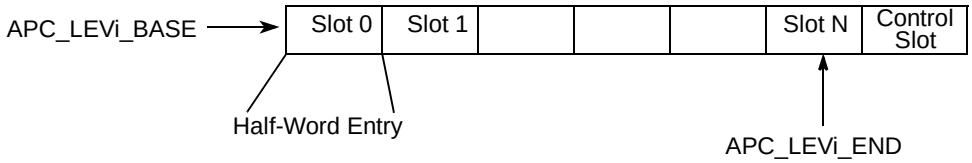


Figure 35-40. The APC Scheduling Table Structure

Slot N+1 is used as a control slot, as shown in Figure 35-41.

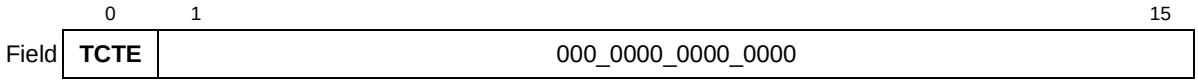


Figure 35-41. Control Slot

Table 35-30 describes the control slot.

Table 35-30. Control Slot Field Description

Bits	Name[1]	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

1 **Boldfaced** entries must be initialized by the user.

35.10.5 ATM Controller Buffer Descriptors (BDs)

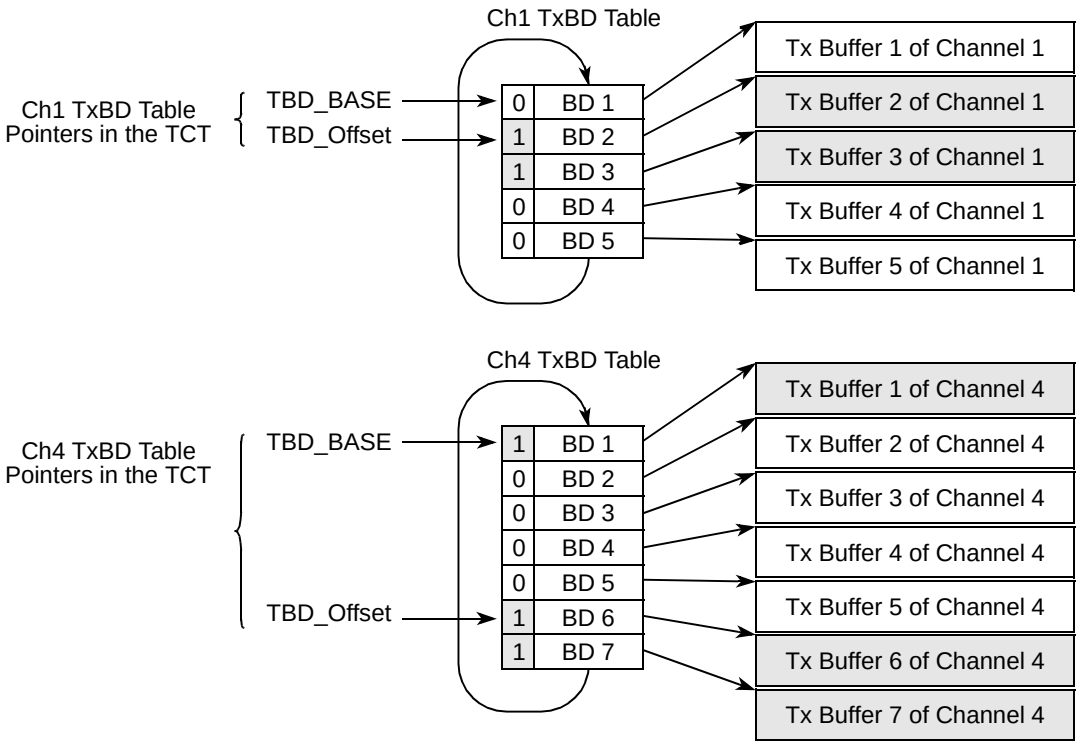
Each ATM channel has separate receive and transmit BD tables. The number of BDs per channel and the size of the buffers is user-defined. The last BD in each table holds a wrap indication. Each BD in the TxBD table points to a buffer to send. At the receive side, the user can choose one of two modes:

- 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 35-42 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 35-42. Transmit Buffers and BD Table Example

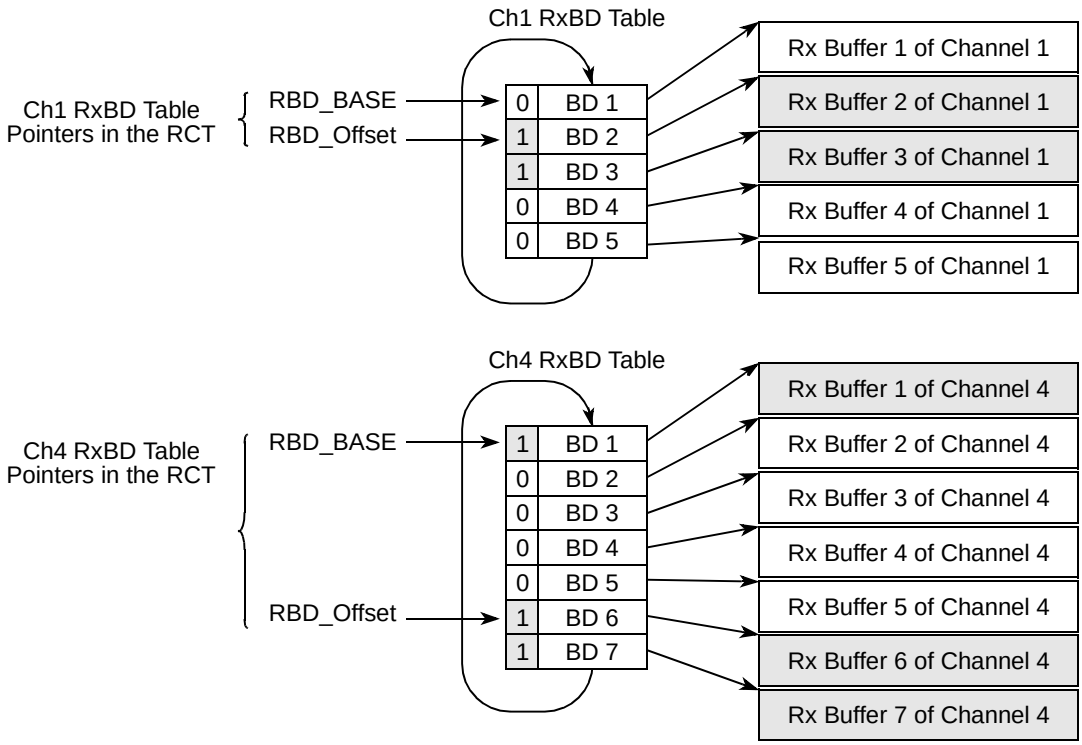
35.10.5.2 Receive Buffer Operation

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 35-43 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.

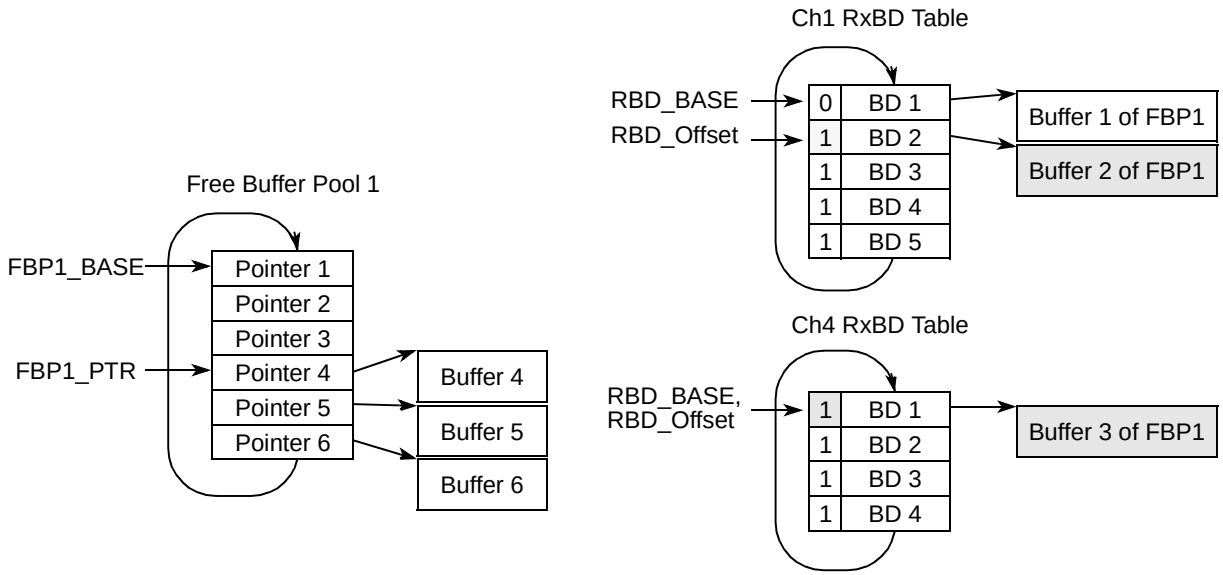
Figure 35-43. Receive Static Buffer Allocation Example

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 35-44 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.



Note: Buffers 2 and 3 are receiving data. After buffer 1 is processed, it can be returned to the pool.

Figure 35-44. Receive Global Buffer Allocation Example

35.10.5.2.3 Free Buffer Pools

As Figure 35-45 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.

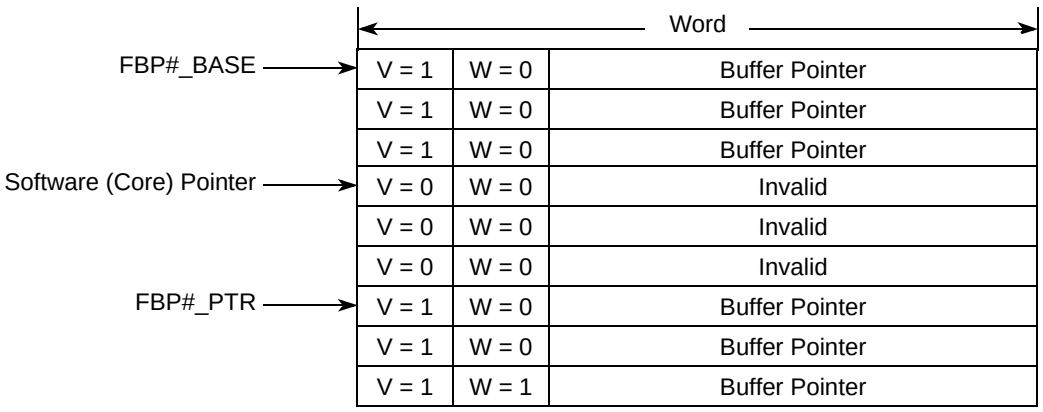


Figure 35-45. Free Buffer Pool Structure

Figure 35-46 describes the structure of a free buffer pool entry.

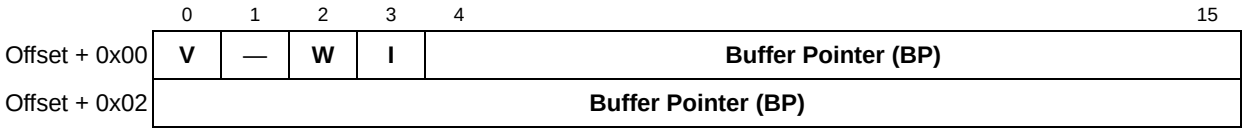


Figure 35-46. Free Buffer Pool Entry

Table 35-31 describes free buffer pool entry fields.

Table 35-31. Free Buffer Pool Entry Field Descriptions

Offset	Bits	Name[1]	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		

[1] **Boldfaced** entries must be initialized by the user.

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 35-32. 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] × 16.

Table 35-32. 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.

Table 35-32. Free Buffer Pool Parameter Table (continued)

Offset [1]	Bits	Name	Description
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.
	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 35-33 describes properties of the ATM receive and transmit buffers.

Table 35-33. 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 35-47 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 35-47. AAL5 RxBD

Table 35-34 describes AAL5 RxBD fields.

Table 35-34. AAL5 RxBD Field Descriptions

Offset	Bits	Name[1]	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 35-34. AAL5 RxBD Field Descriptions (continued)

Offset	Bits	Name[1]	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.

1 **Boldfaced** entries must be initialized by the user.

35.10.5.5 AAL1 RxBD

Figure 35-48 shows the AAL1 RxBD.

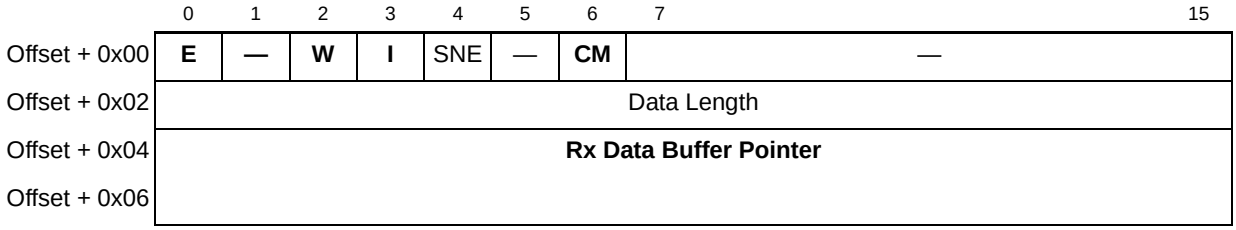


Figure 35-48. AAL1 RxBD

Table 35-35 describes AAL1 RxBD fields.

Table 35-35. AAL1 RxBD Field Descriptions

Offset	Bits	Name[1]	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.

[1] **Boldfaced** entries must be initialized by the user.

35.10.5.6 AAL0 RxBD

Figure 35-49 shows the AAL0 RxBD.

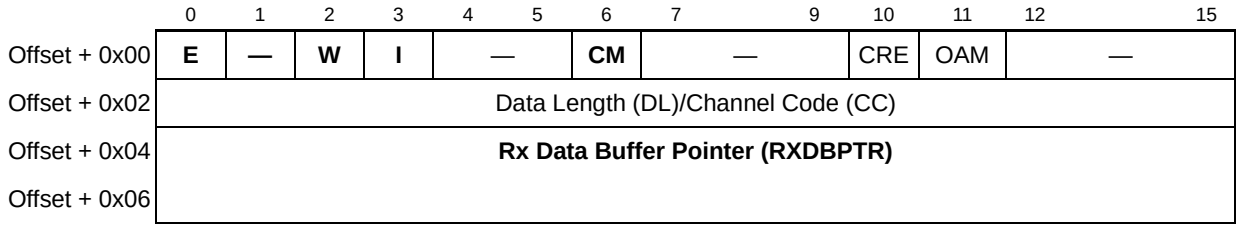


Figure 35-49. AAL0 RxBD

Table 35-36 describes AAL0 RxBD fields.

Table 35-36. AAL0 RxBD Field Descriptions

Offset	Bits	Name[1]	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.

1 **Boldfaced** entries must be initialized by the user.

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 35-50. 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 35-50. User-Defined Cell—RxBD Extension

35.10.5.8 AAL5 TxBDs

Figure 35-51 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 35-51. AAL5 TxBD

Table 35-37 describes AAL5 TxBD fields.

Table 35-37. AAL5 TxBD Field Descriptions

Offset	Bits	Name[1]	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.

Table 35-37. AAL5 TxBD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
0x00	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.

1 **Boldfaced** entries must be initialized by the user.

35.10.5.9 AAL1 TxBDs

Figure 35-52 shows the AAL1 TxBD.

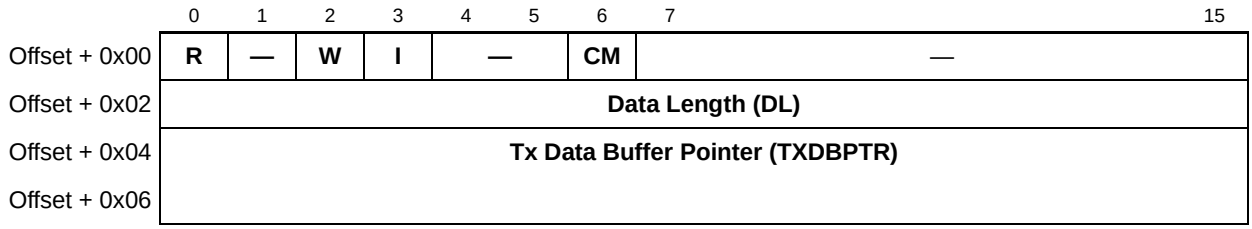


Figure 35-52. AAL1 TxBD

Table 35-38 describes AAL1 TxBD fields.

Table 35-38. AAL1 TxBD Field Descriptions

Offset	Bits	Name[1]	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.

[1] **Boldfaced** entries must be initialized by the user.

35.10.5.10 AAL0 TxBDs

Figure 35-53 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.

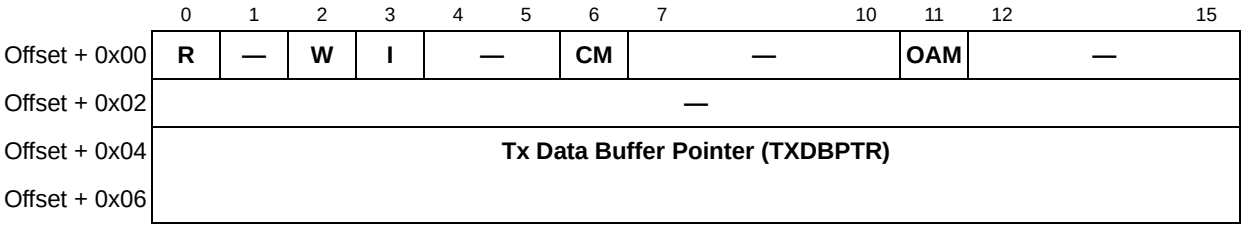


Figure 35-53. AAL0 TxBDs

Table 35-39 describes AAL0 TxBD fields.

Table 35-39. AAL0 TxBD Field Descriptions

Offset	Bits	Name[1]	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.

[1] **Boldfaced** entries must be initialized by the user.

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 35-54. 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 35-54. 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 35-55.

	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 35-55. AAL1 Sequence Number (SN) Protection Table

35.10.7 UNI Statistics Table

The UNI statistics table, shown in Table 35-40, 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 35-40. 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.

NOTE

Ensure that the transmit INTQs and the receive INTQs are programmed as separate queues.

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.

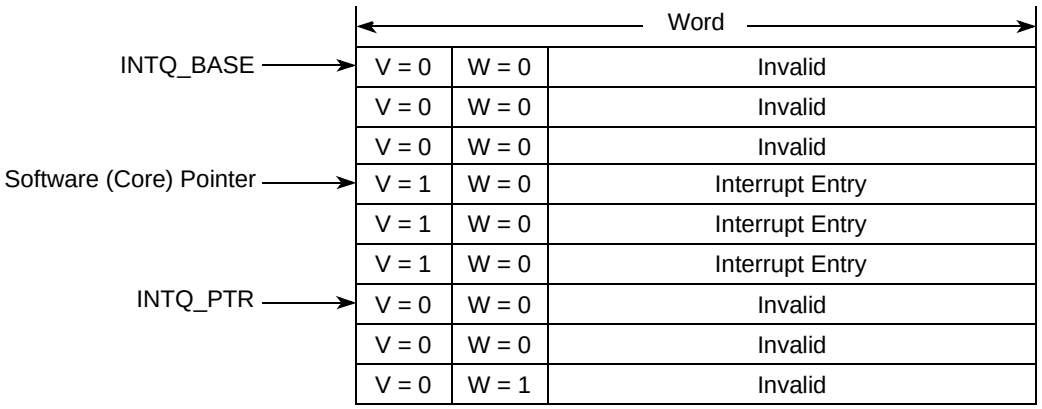


Figure 35-56. Interrupt Queue Structure

35.11.2 Interrupt Queue Entry

Each one-word interrupt queue entry provides detailed interrupt information to the host. Figure 35-57 shows an entry.

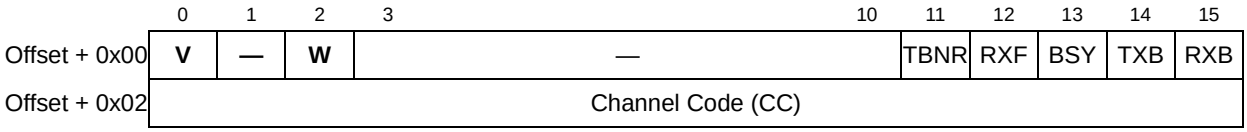


Figure 35-57. Interrupt Queue Entry

Table 35-41 describes interrupt queue entry fields.

Table 35-41. Interrupt Queue Entry Field Description

Offset	Bits	Name[1]	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.

1 **Boldfaced** entries must be initialized by the user.

35.11.3 Interrupt Queue Parameter Tables

The interrupt queue parameters are held in parameter tables in the dual-port RAM; see Table 35-42. 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 35-42. Interrupt Queue Parameter Table

Offset [1]	Name[2]	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.
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.

2 **Boldfaced** entries must be initialized by the user.

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 35-58.

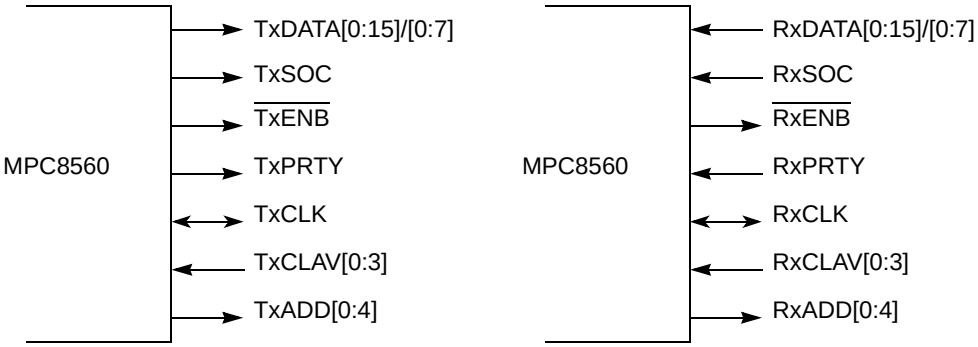


Figure 35-58. UTOPIA Master Mode Signals

Table 35-43 describes UTOPIA master mode signals.

Table 35-43. 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.
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.
$\overline{\text{RxENB}}$	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 35-59.

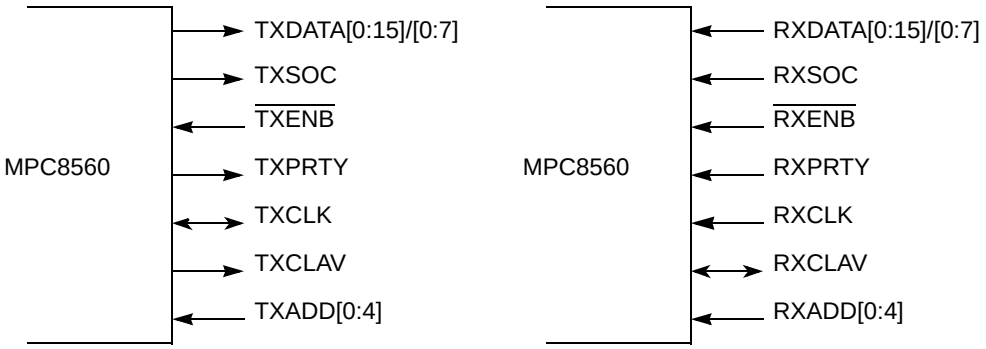


Figure 35-59. UTOPIA Slave Mode Signals

Table 35-44 describes UTOPIA slave mode signals.

Table 35-44. 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.

Table 35-44. UTOPIA Slave Mode Signals

Signal	Description
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 35-45.

Table 35-45. 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 35-60, 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.

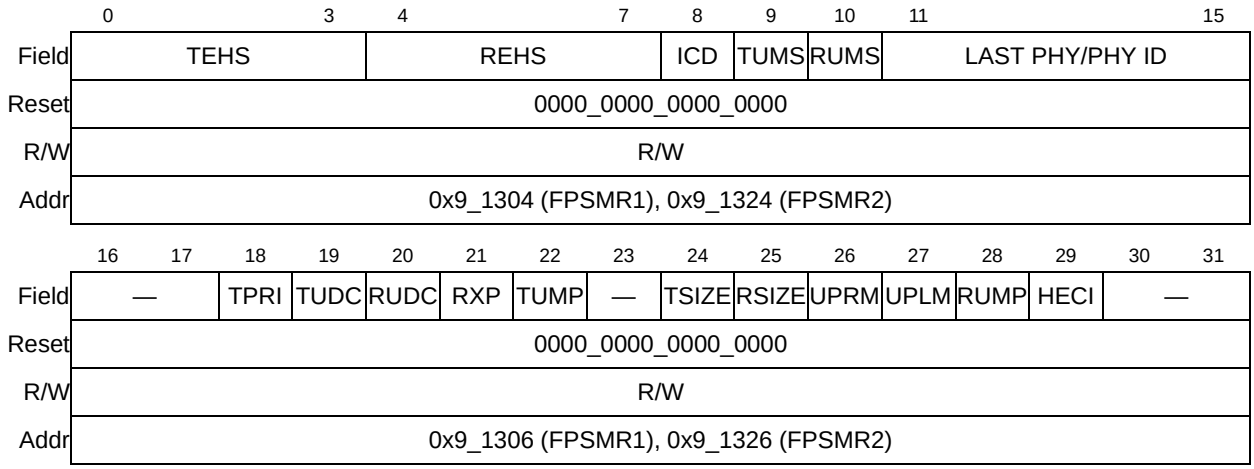


Figure 35-60. FCC ATM Mode Register (FPSMR)

Table 35-46 describes FPSMR fields.

Table 35-46. 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.

Table 35-46. FCC ATM Mode Register (FPSMR) (continued)

Bits	Name	Description
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.
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.

Table 35-46. FCC ATM Mode Register (FPSMR) (continued)

Bits	Name	Description
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 35-61 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 35-61. ATM Event Register (FCCE)/FCC Mask Register (FCCM)

Table 35-47 describes FCCE fields.

Table 35-47. 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.

35.13.4 FCC 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.

35.13.5 FCC 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

Figure 35-62. FCC Transmit Internal Rate Port Enable Register (FIRPER)

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 35-62. FCC Transmit Internal Rate Port Enable Register (FIRPER)

Table 35-48 describes FIRPER*x* fields.

Table 35-48. FIRPER*x* Field Descriptions (TIREM = 1)

Bits	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.

35.13.6 FCC 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 35-63. FCC Internal Rate Event Register (FIRER)

Table 35-49 describes FIRER*x* fields.

Table 35-49. FIRER*x* Field Descriptions (TIREM=1)

Bits	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.

35.13.7 FCC 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 35-64 and Figure 35-65, 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 35-64. FCC Internal Rate Selection Register HI (FIRSR*x*_HI)

Table 35-50 describes FIRSR*x*_HI fields.

Table 35-50. IRSR*x*_HI Field Descriptions (TIREM=1)

Bits	Name	Description
0–31	GSy	Group select for PHY y 00 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP0. 01 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP1. 10 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP2. 11 The 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 35-65. FCC Internal Rate Selection Register LO (FIRSR*x*_LO)

Table 35-51 describes FIRSR*x*_LO fields.

Table 35-51. FIRSR*x*_LO Field Descriptions (TIREM=1)

Bits	Name	Description
0-29	GSy	Group select for PHY y 00 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP0. 01 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP1. 10 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP2. 11 The transmit internal rate for PHY address y is controlled by FTIRRx_GRP3.
30–31	—	Reserved, should be cleared.

35.13.8 FCC 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 35-66, 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 35-66. FCC Transmit Internal Rate Register (FTIRR)

Table 35-52 describes FTIRR*x* fields.

Table 35-52. FTIRR*x* Field Descriptions

Bits	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 35-67 shows how transmit clocks are determined.

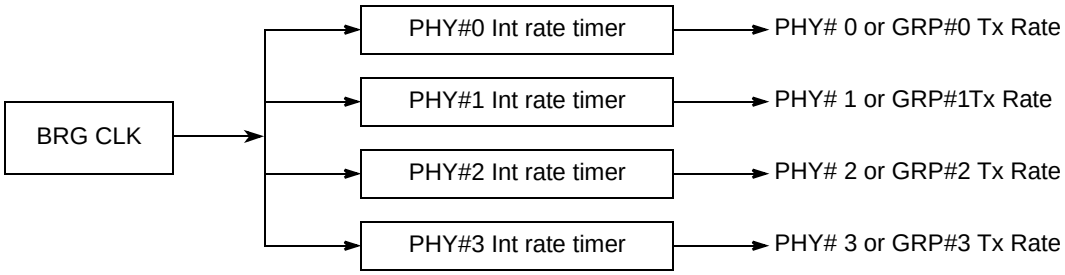


Figure 35-67. FCC Transmit Internal Rate Clocking

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 35-68.

Offset	0–4	5	6–10	11–12	13–15
0x86	—	CTB	PHY#	ACT	PRI
0x88	Channel Code (CC)				
0x8A	BT				

Figure 35-68. COMM_INFO Field

Table 35-53 describes COMM_INFO fields.

Table 35-53. 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 35-69.

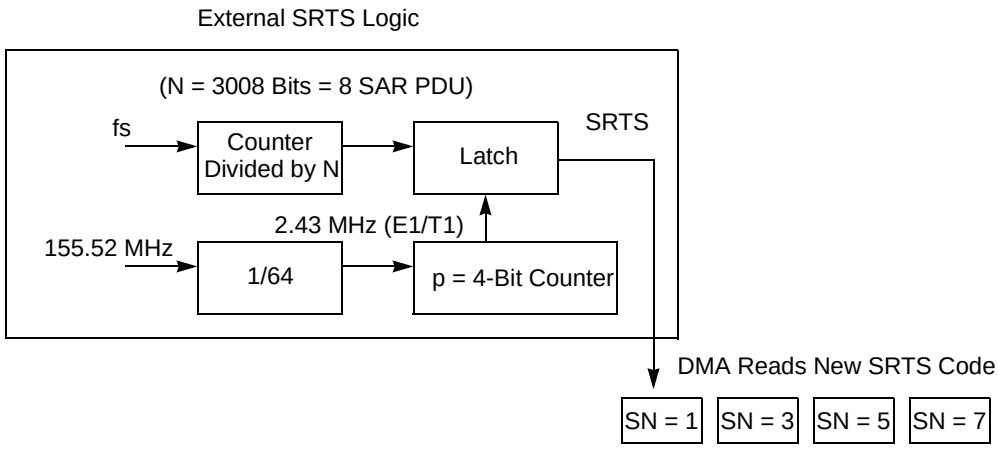


Figure 35-69. AAL1 SRTS Generation Using External Logic

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 35-70.

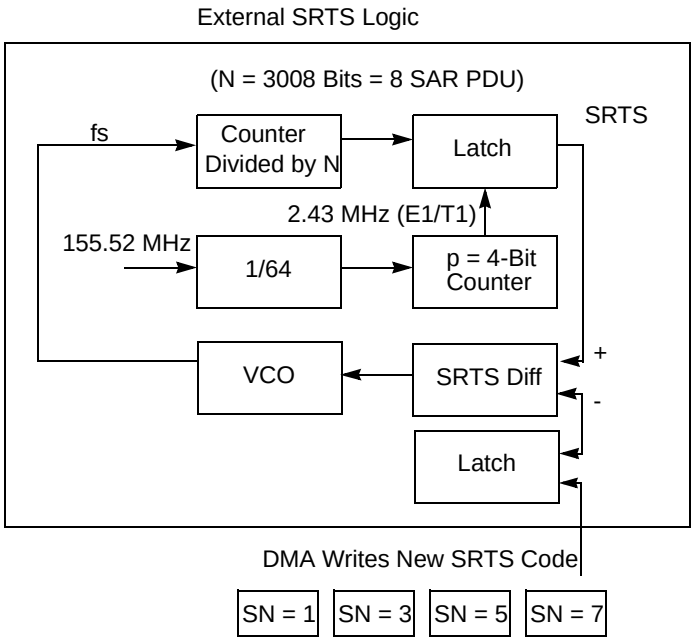


Figure 35-70. AAL1 SRTS Clock Recovery Using External Logic

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 (for example, 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 2 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 is transferred between the ATM and MCC automatically without CPU intervention.
    - In case of pre-underrun, the MCC starts 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 is 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.
      - Supports 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.

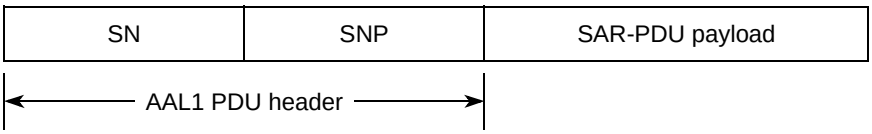


Figure 36-1. AAL1 Transmit Cell Format

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.

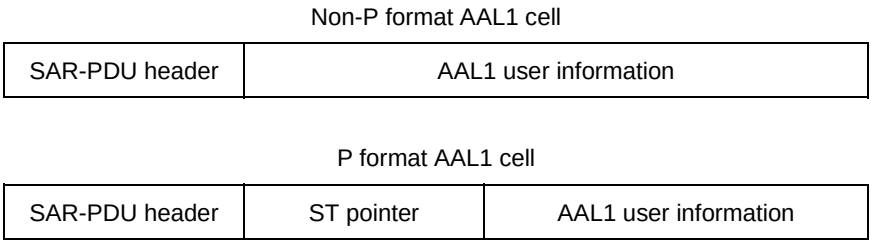


Figure 36-2. Non-P Format AAL1 Cell

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.

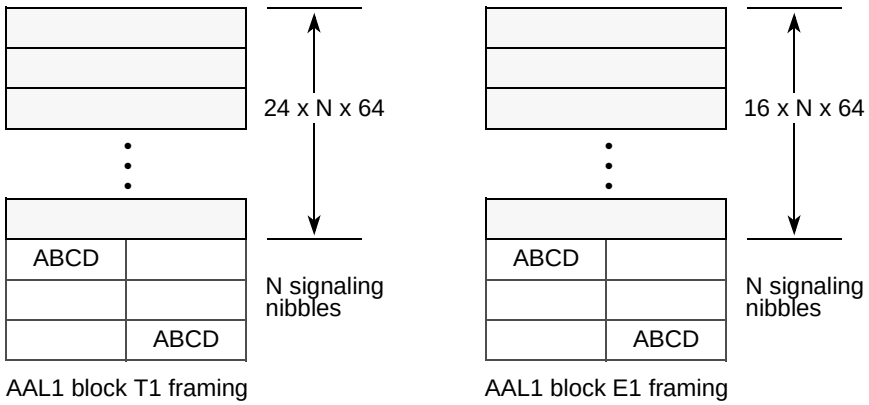
The MPC8560 supports synchronous residual time stamp (SRTS) generation using an external PLL. If this mode is enabled, the MPC8560 reads the SRTS code from external logic and inserts it into four outgoing cells. See Section 36.17, “SRTS Generation and Clock Recovery Using External SRTS Logic.”

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 with 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) nibbles 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) 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 superframe, which is the first BD after the one marked with EOSF (end of superframe). When this BD is ready and the adaptive counter reaches the ATM_Start threshold, the receiver's write pointer is no 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 superframe). (See Section 36.5, "ATM-to-TDM Adaptive Slip Control," and Section 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 Section 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.

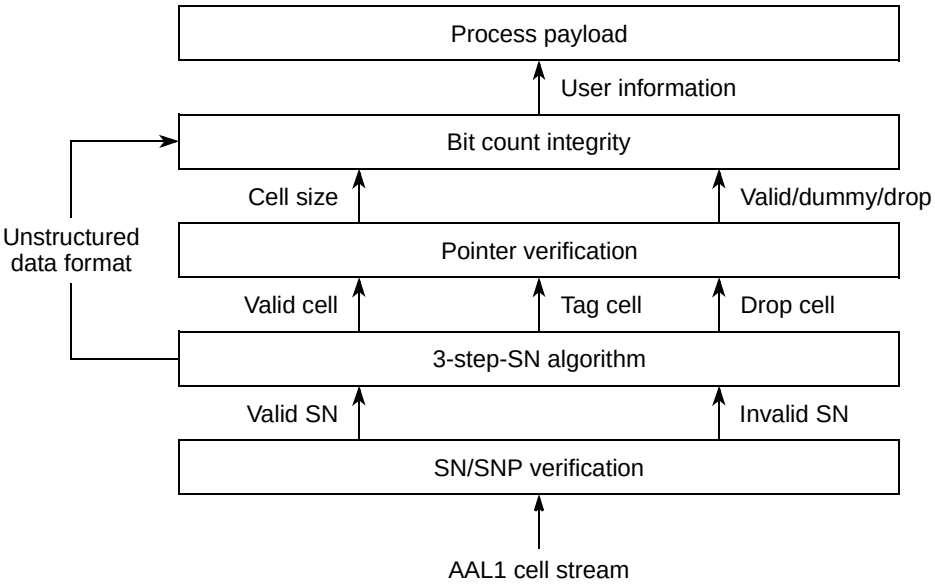


Figure 36-4. AAL1 Receiver Data Flow

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 36-5), 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. In 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.

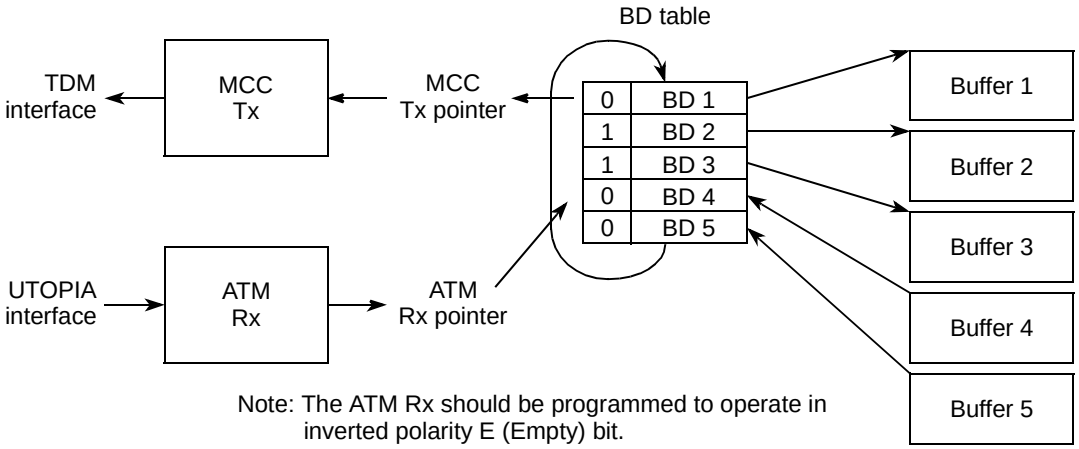


Figure 36-5. ATM-to-TDM Interworking

36.4.1.2 TDM-to-ATM

When going from TDM to ATM (depicted in Figure 36-6), 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 Section 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.

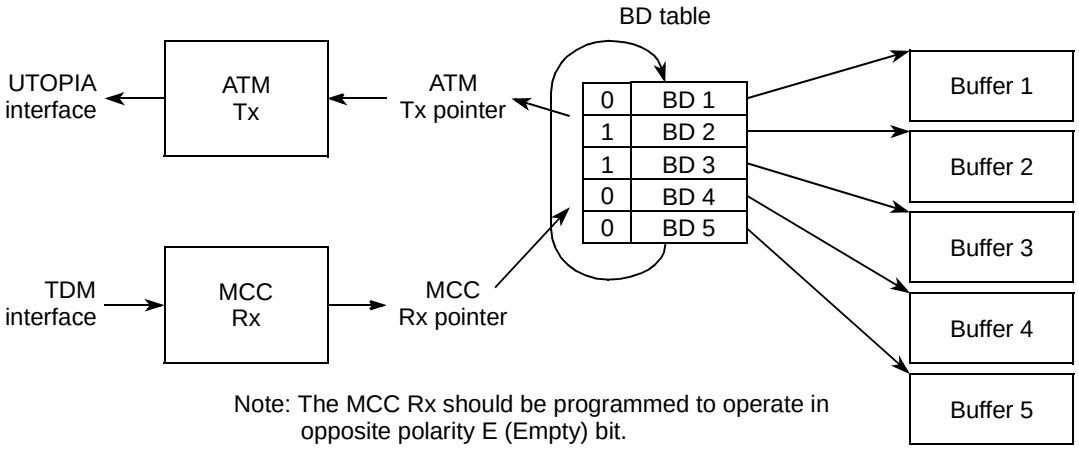


Figure 36-6. TDM-to-ATM Interworking

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 Section 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 no 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 superframe). (See Section 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:

- 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.”)
- 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 Section 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

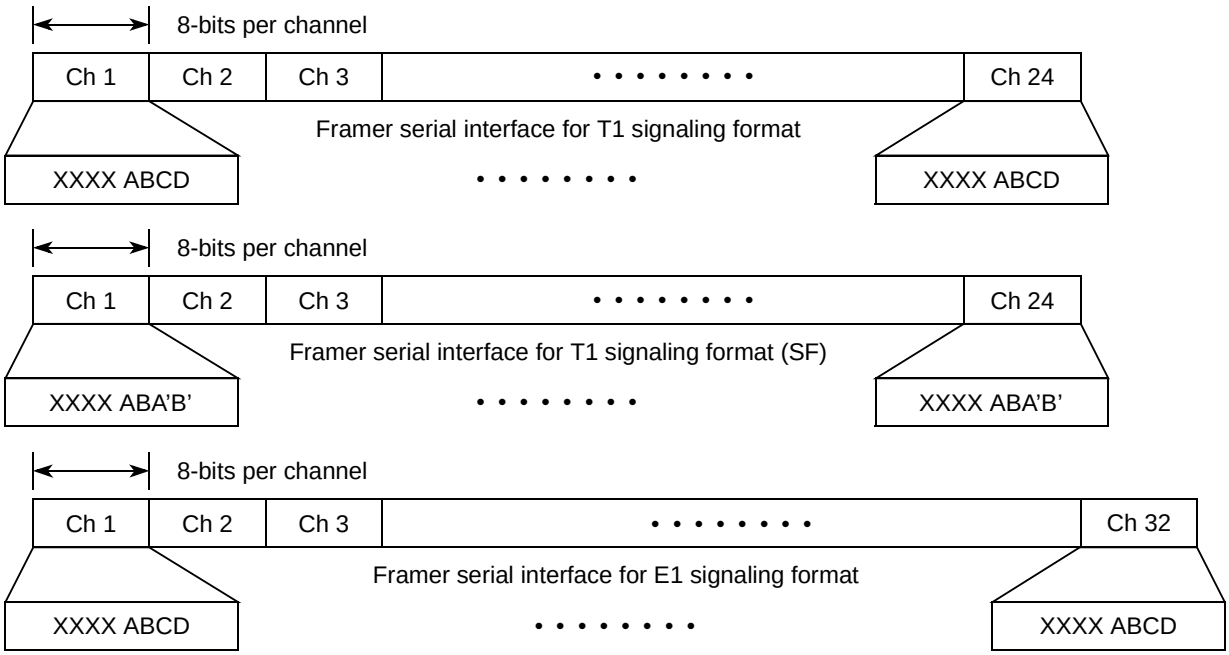
BD table) should be used by the ATM controller to route both the receive and transmit data. For information about ATM buffers see Section 36.11, “Buffer Descriptors.”

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 Section 36.4.7.1, “CAS Routing Table.”

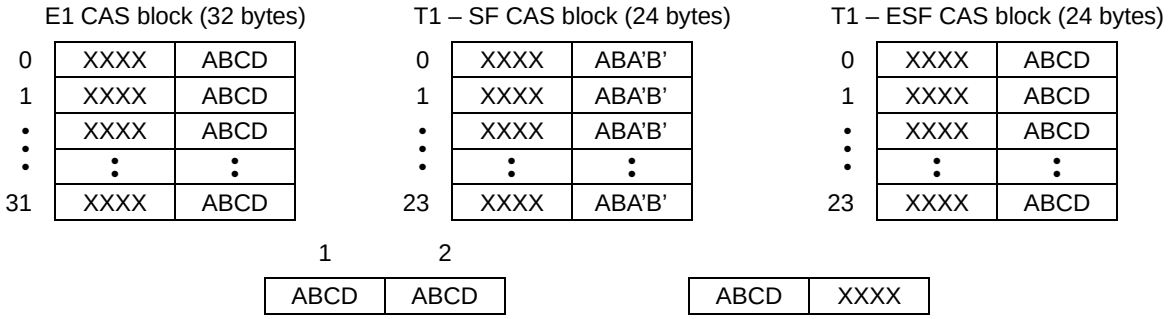


Note: The MCC should capture the signaling data on the last frame of a superframe. See Section 1.4.7.2, “TDM-to-ATM CAS Support.”

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.



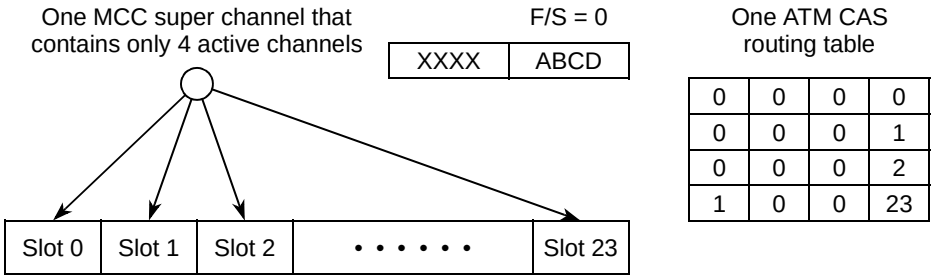
Note: CAS block resides in the internal RAM. To allow maximum flexibility in working with external framers, the signaling nibble can occupy the first or second nibble in the CAS entry.

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 Section 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).

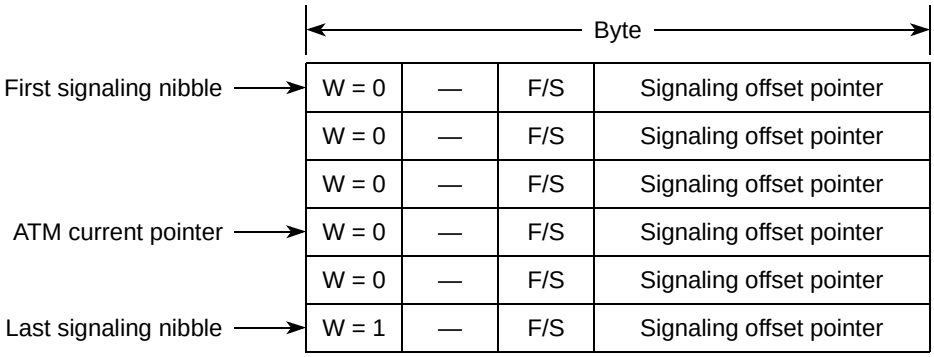


Figure 36-10. AAL1 CAS Routing Table (CRT)

Figure 36-11 describes the structure of a CAS routing table entry.

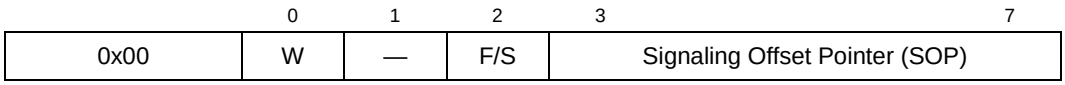


Figure 36-11. AAL1 CAS Routing Table Entry

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.

Table 36-1. CAS Routing Table Entry Field Descriptions

Bits	Name	Description
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 superframe (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 superframes. The last buffer of the superframe is marked as the end of a superframe (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. See Figure 36-12.

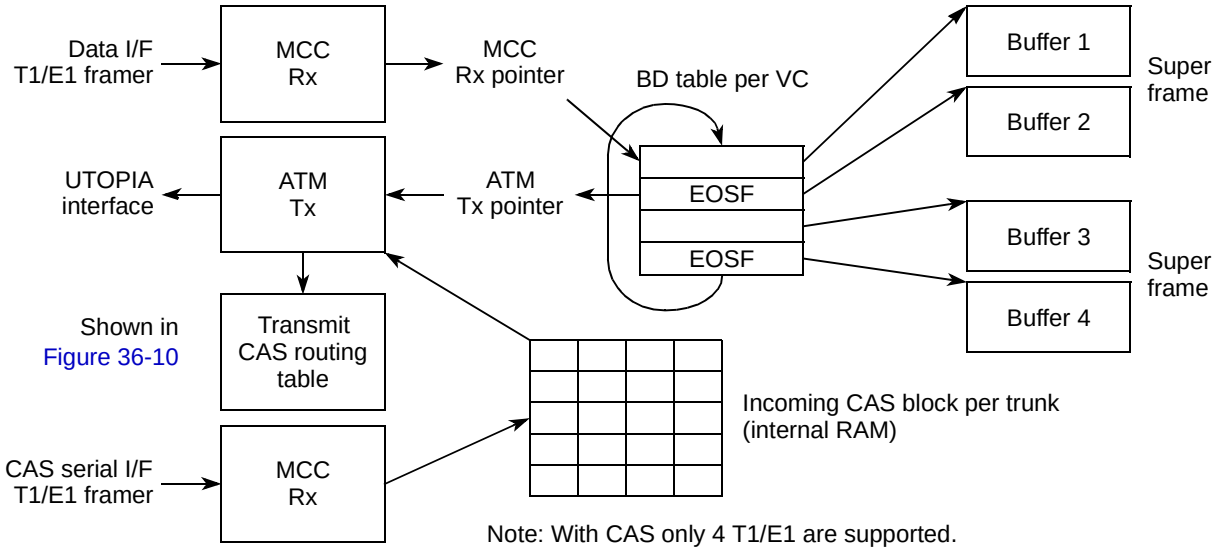


Figure 36-12. CAS Flow TDM-to-ATM

The CAS block is automatically written to internal RAM by the MCC receiver using a separate TDM. When a superframe is received the MCC should be triggered with a superframe (multi-frame) SYNC from the external framer. The incoming CAS block should be captured by the MCC only once for each superframe (on the last frame).

External logic may be used to convert the framer superframe 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 frame reassembly process is shown in Figure 36-13. During reassembly, the AAL1 receiver unpacks the signaling information from the end of an AAL1 superframe (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, and 3-step-SN algorithm), operate normally. Each common (MCC, ATM) BD table should point to buffers that can store a whole number of superframes. The last buffer of the superframe 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.

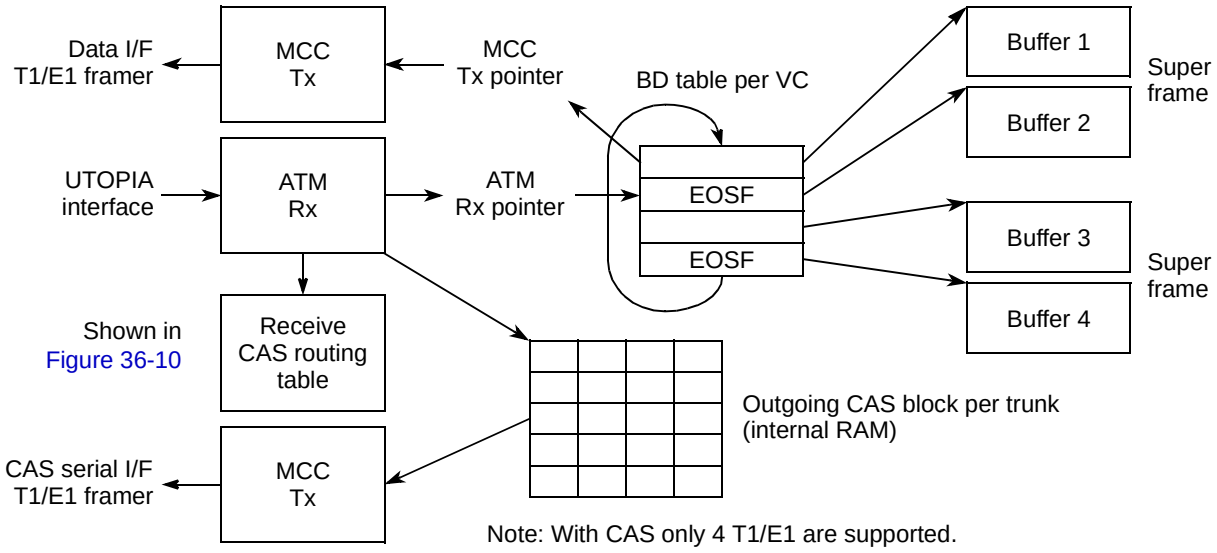


Figure 36-13. CAS Flow ATM-to-TDM

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 nibbles 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 that 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 35.11.3, “Interrupt Queue Parameter Tables.”

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 Figure 36-14). 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.

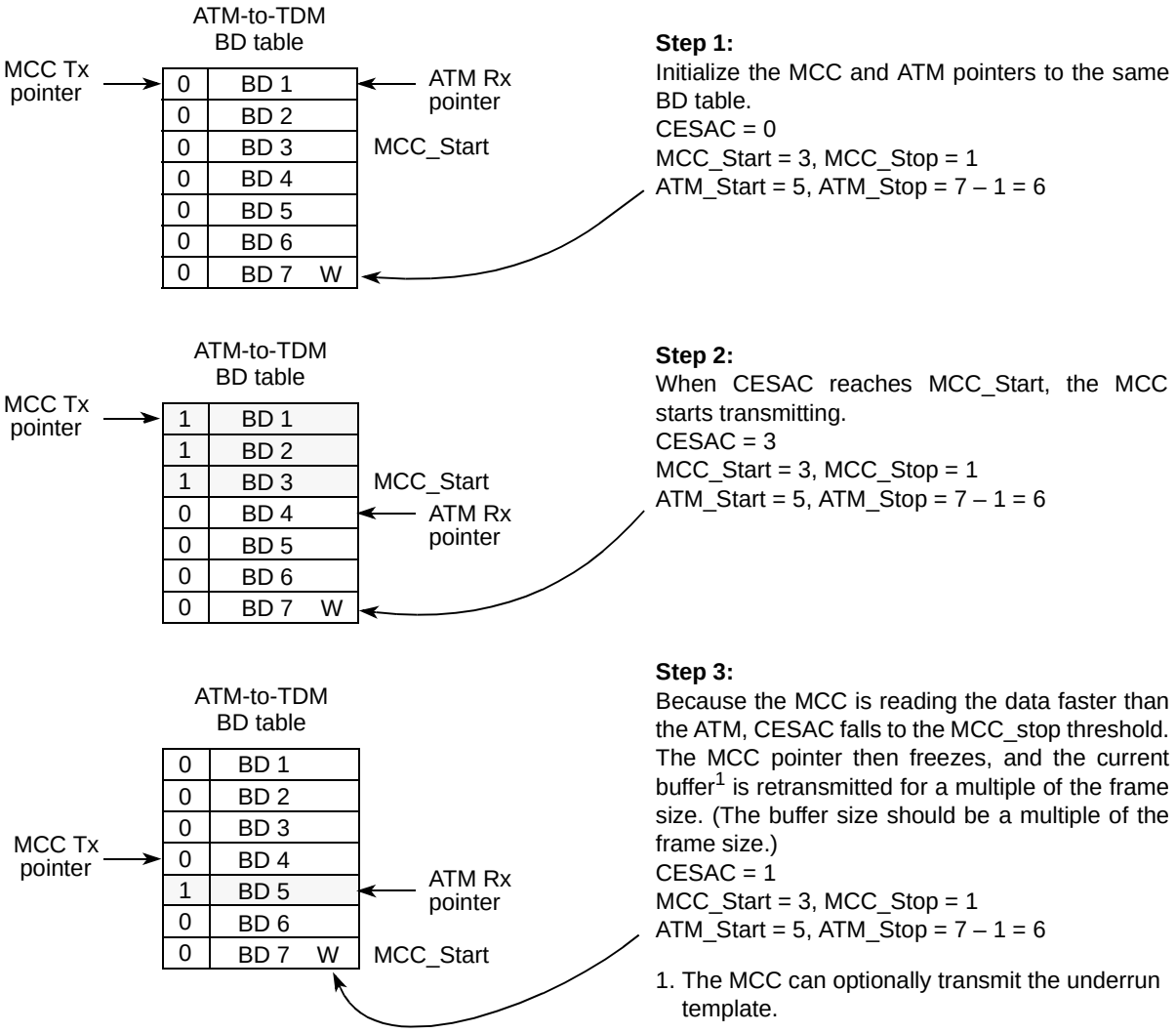


Figure 36-14. Pre-Underrun Sequence

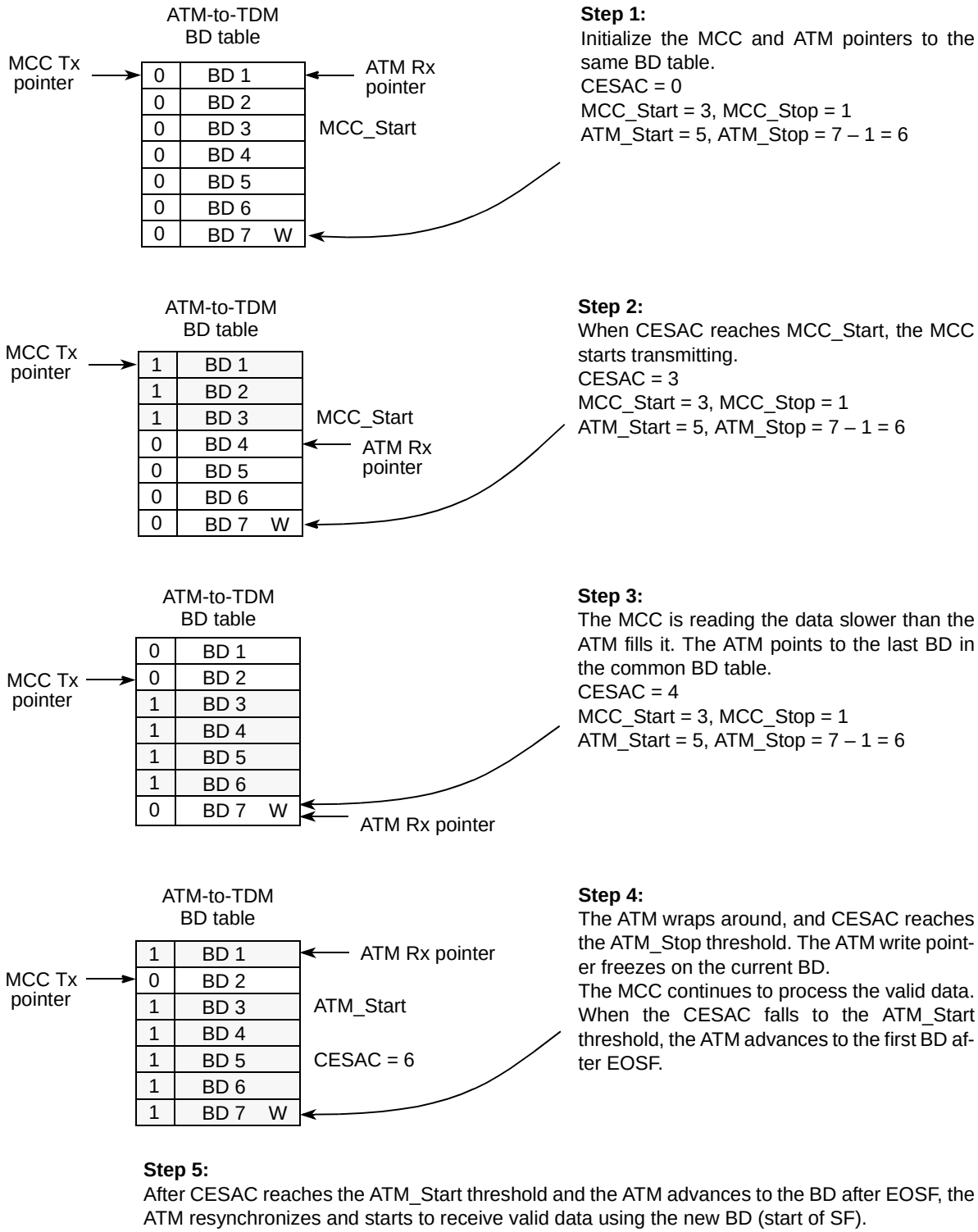
36.5.2 Pre-Overrun

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 superframe).

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.



Step 1:
Initialize the MCC and ATM pointers to the same BD table.
CESAC = 0
MCC_Start = 3, MCC_Stop = 1
ATM_Start = 5, ATM_Stop = 7 – 1 = 6

Step 2:
When CESAC reaches MCC_Start, the MCC starts transmitting.
CESAC = 3
MCC_Start = 3, MCC_Stop = 1
ATM_Start = 5, ATM_Stop = 7 – 1 = 6

Step 3:
The MCC is reading the data slower than the ATM fills it. The ATM points to the last BD in the common BD table.
CESAC = 4
MCC_Start = 3, MCC_Stop = 1
ATM_Start = 5, ATM_Stop = 7 – 1 = 6

Step 4:
The ATM wraps around, and CESAC reaches the ATM_Stop threshold. The ATM write pointer freezes on the current BD.
The MCC continues to process the valid data. When the CESAC falls to the ATM_Start threshold, the ATM advances to the first BD after EOSF.

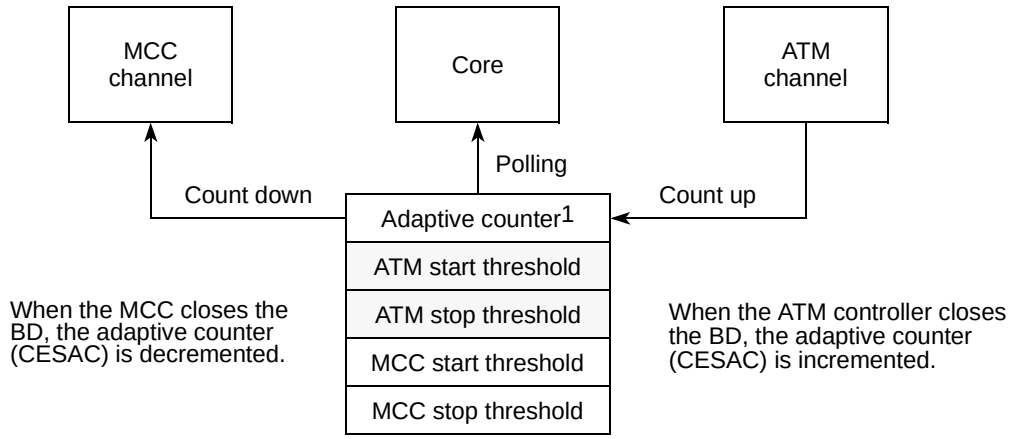
Step 5:
After CESAC reaches the ATM_Start threshold and the ATM advances to the BD after EOSF, the ATM resynchronizes and starts to receive valid data using the new BD (start of SF).

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 36-17) 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	—	

Figure 36-17. CES Adaptive Threshold Table

Table 36-2 describes CES adaptive threshold table fields.

Table 36-2. 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. 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 Section 36.5, “ATM-to-TDM Adaptive Slip Control.”) MSTP 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. 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.

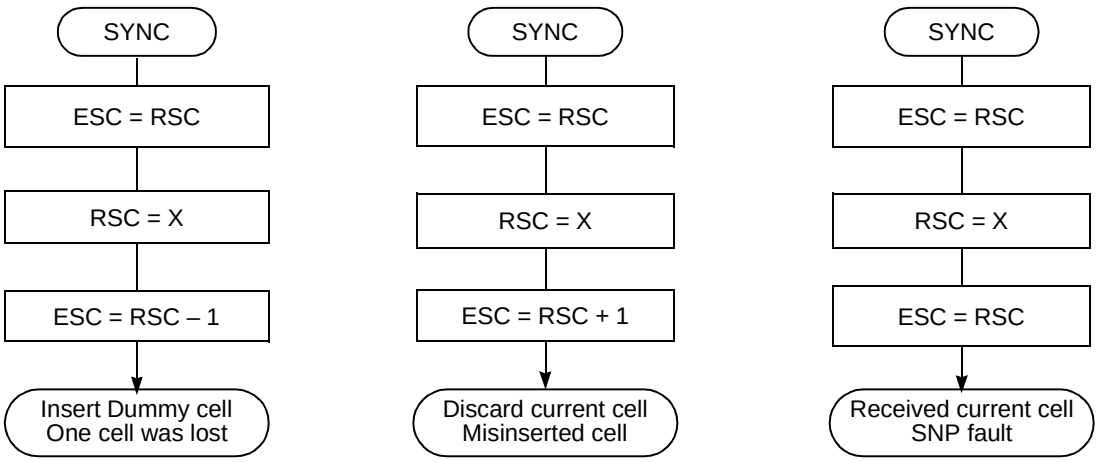
36.6 3-Step-SN Algorithm

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 36-18. Recoverable Sync Fail Sequence Options

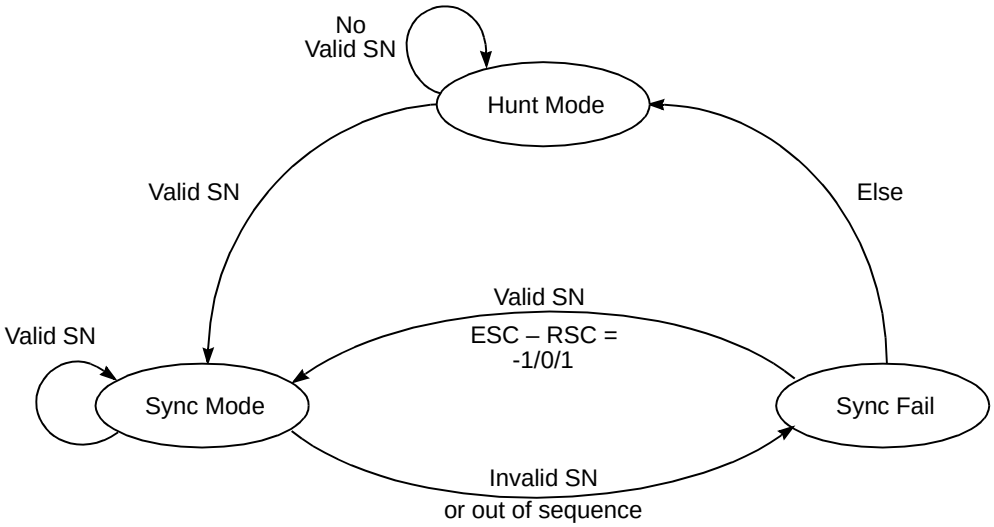


Figure 36-19. 3-Step-SN Algorithm

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.

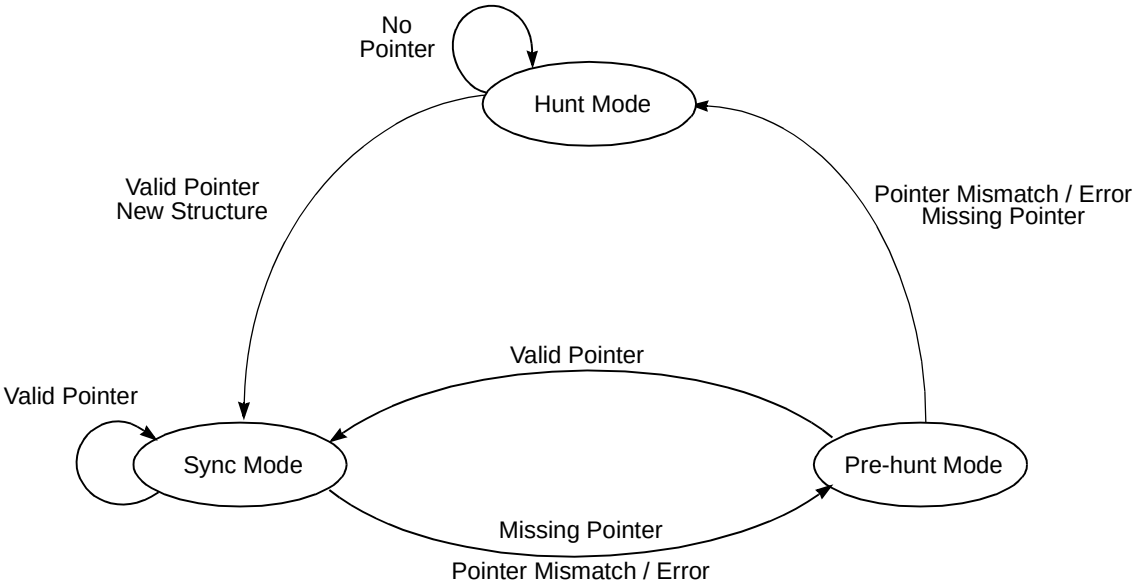


Figure 36-20. Pointer Verification Mechanism

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 36-3. 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)
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.

Table 36-3. AAL1 Field Descriptions (continued)

Offset	Name	Width	Description
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).”
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	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 prescalar (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.

Table 36-3. AAL1 Field Descriptions (continued)

Offset	Name	Width	Description
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 36-4. 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 Section 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.

Table 36-4. AAL1 CES Parameters (continued)

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 Section 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 Section 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 36-21 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 36-21. 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 36-5 describes RCT fields.

Table 36-5. 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 36-5. 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 36-5. 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 36-22 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 36-22. AAL1 Protocol-Specific RCT

Table 36-6 describes AAL1 protocol-specific RCT fields.

Table 36-6. 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.

Table 36-6. AAL1 Protocol-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
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 Section 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
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.

Table 36-6. AAL1 Protocol-Specific RCT Field Descriptions (continued)

Offset	Bits	Name	Description
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 Figure 36-15 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 36-23 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 36-23. Transmit Connection Table (TCT) Entry

Table 36-7 describes general TCT fields.

Table 36-7. 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 36-7. 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 36-7. 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 36-24 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 36-24. AAL1 Protocol-Specific TCT

Table 36-8 describes AAL1 protocol-specific TCT fields.

Table 36-8. 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 36.17, “SRTS Generation and Clock Recovery Using External SRTS Logic.”) 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.

Table 36-8. AAL1-Specific TCT Field Descriptions (continued)

Offset	Bits	Name	Description
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 Section 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 36-25 shows the outgoing CAS block status register (OCASSR).

	15	14	13	12	11	10	9	8	7	0
Field	MCASB0	MCASB1	MCASB2	MCASB3	MCASB4	MCASB5	MCASB6	MCASB7	—	
Reset	0x_0000									
R/W	R/W									
Offset	0xD0									

Figure 36-25. 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 36-9 describes OCASSR fields.

Table 36-9. OCASSR Field Descriptions

Bits	Name	Description
0–7	—	Reserved, should be cleared during initialization.
8–15	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."

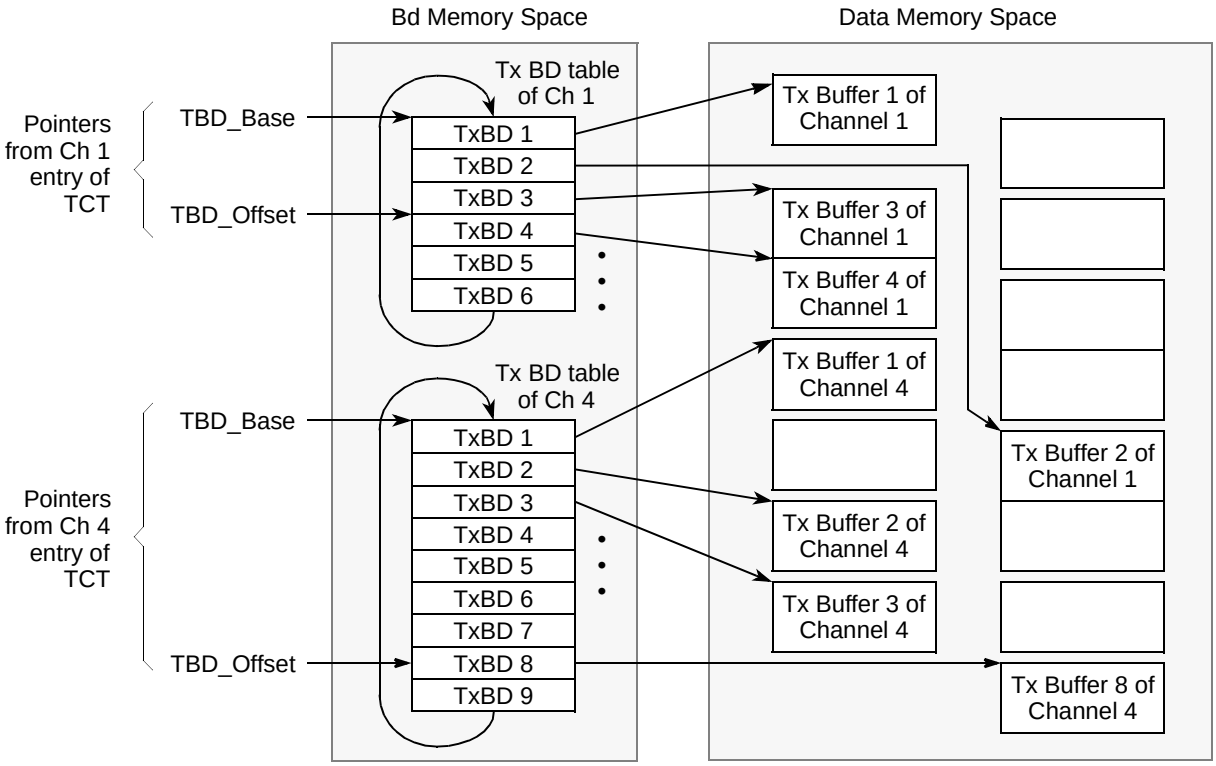


Figure 36-26. Transmit Buffers and BD Table Example

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."

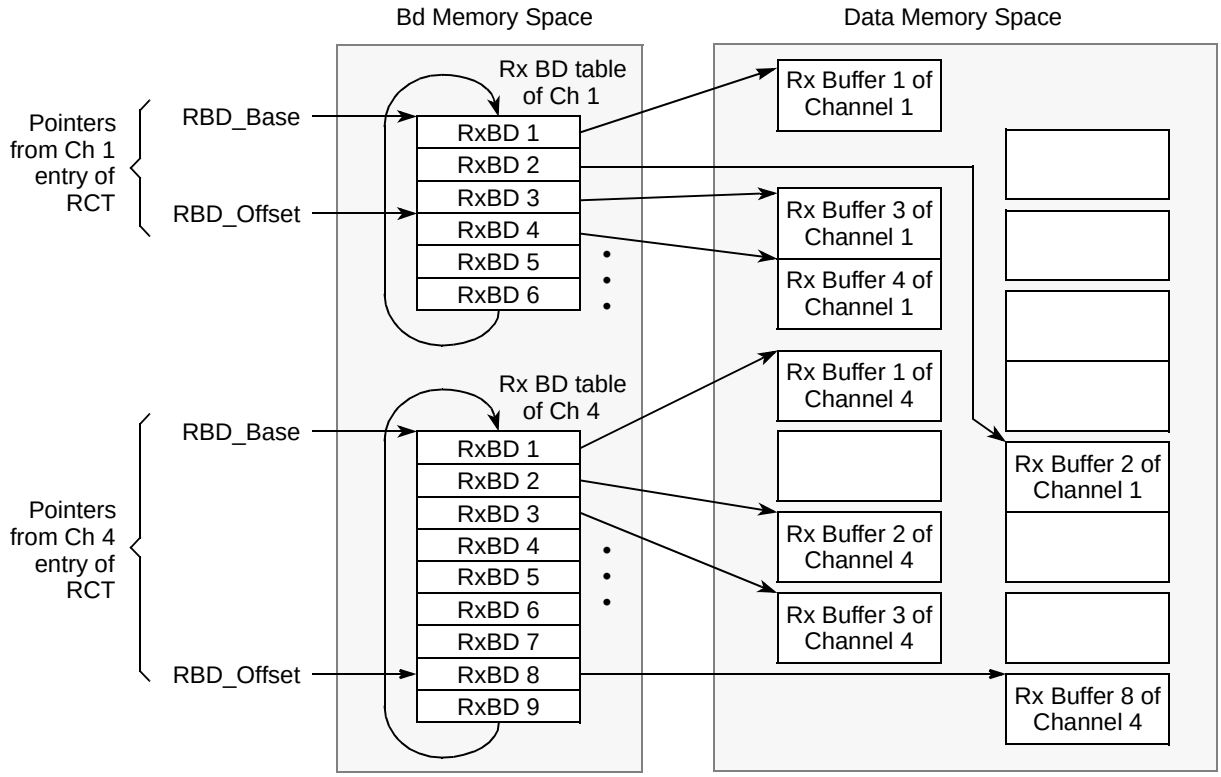


Figure 36-27. Receive Buffers and BD Table Example

36.12 ATM Controller Buffers

Table 36-10 describes properties of the ATM receive and transmit buffers.

Table 36-10. 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
AAL1	Multiple of 8 octets	No requirement	Multiple of 8 octets	No requirement
AAL0	52-64 octets.	Burst-aligned	52–64 octets.	No requirement

36.12.1 AAL1 RxBD

Figure 36-28 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 36-28. AAL1 RxBD

Table 36-11 describes AAL1 RxBD fields.

Table 36-11. AAL1 RxBD Field Descriptions

Offset	Bits	Name[1]	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 superframe. 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 superframe in the common (MCC, ATM) BD table. See Section 36.4.6, “Channel Associated Signaling (CAS) Support.”
	10–15	—	—

Table 36-11. AAL1 RxBD Field Descriptions

Offset	Bits	Name[1]	Description
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.

1 **Boldfaced** entries must be initialized by the user.

36.12.2 AAL1 TxBDs

Figure 36-29 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 36-29. AAL1 TxBD

Table 36-12 describes AAL1 TxBD fields.

Table 36-12. AAL1 TxBD Field Descriptions

Offset	Bits	Name[1]	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	—	—

Table 36-12. AAL1 TxBD Field Descriptions (continued)

Offset	Bits	Name[1]	Description
	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 superframe. 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 superframe in the common (MCC, ATM) BD table. See Section 36.4.6, “Channel Associated Signaling (CAS) Support.”
	10–15	—	—
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.

1 **Boldfaced** entries must be initialized by the user.

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 35.11, “ATM Exceptions.”

36.13.1 AAL1 Interrupt Queue Entry

Figure 36-30 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 36-30. AAL1 Interrupt Queue Entry

Table 36-13 describes interrupt queue entry fields.

Table 36-13. AAL1 Interrupt Queue Entry Field Descriptions

Offset	Bits	Name[1]	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	**—**	—
	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 Figure 36-15. 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 Figure 36-15. 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.

1 **Boldfaced** entries must be initialized by the user.

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 36-14.

Table 36-14. AAL1 Sequence Number (SN) Protection Table

Local Offset	0 15
0x00	0x0000
0x02	0x0007
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 36-15, 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 36-15. 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 36-16, 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 36-16. 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 Figure 36-15.
0x02	Rx_AAL1_MISS	Hword	16-bit cyclic counter. Counts the number of AAL1 misinserted events. See Figure 36-15.
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 Figure 36-15.
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 36-31.

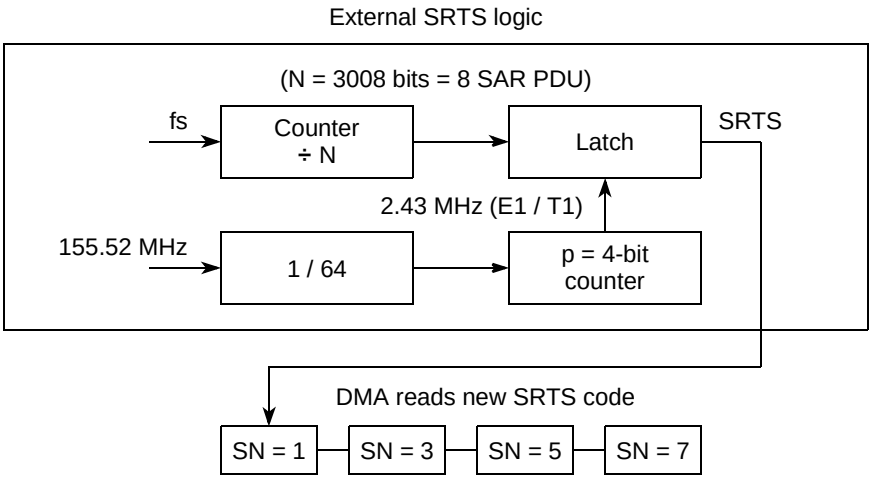


Figure 36-31. AAL1 SRTS Generation Using External Logic

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 36-32.

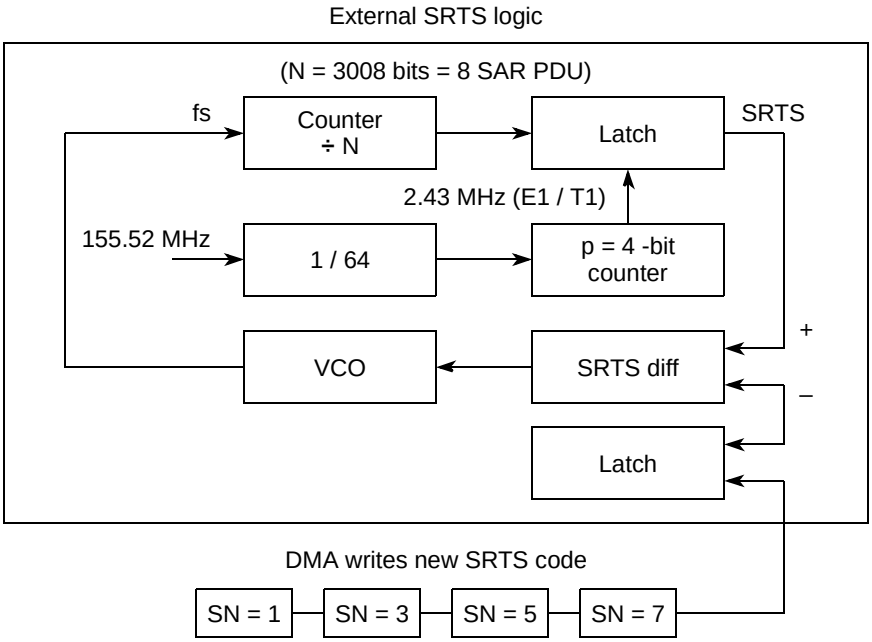


Figure 36-32. AAL1 SRTS Clock Recovery Using External Logic

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 36-18.

Table 36-17. 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 Section 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 36-33 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 36-33. Channel Mode Register (CHAMR)—CES Mode

The CHAMR in CES mode fields are described in Table 36-18.

Table 36-18. CHAMR Field Descriptions—CES Mode

Bits	Name	Description
0	MODE	Channel mode. Selects either HDLC or transparent mode. Must be cleared for CES operation. 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	—	Reserved. Must be set.
4	EP	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.
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
01	Slot	Slot	The first data is sent/received in the slot defined in the slot assignment table (for super channels only)
10	8-bit	None	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
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).

8–9	—	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 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

Table 36-18. CHAMR Field Descriptions—CES Mode (continued)

Bits	Name	Description
13	CESM	Circuit emulation service mode. 0 Normal mode 1 CES mode
14	UDC	User-defined cell support. 0 User-defined ATM cells are not supported. 1 User-defined ATM cells are supported.
15	UTM	Underrun template mode. 0 Retransmit the last buffer. 1 Send 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 36-34 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 36-34. MCC Interrupt Table Entry with CES Slip Indicators and INTMSK

The two MCC transmitter slip indicators are described in Table 36-19.

Table 36-19. 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 superframe synchronization for the data flow in ATM-to-TDM and TDM-to-ATM interworking. However, for the signaling flow, the

MPC8560 uses the superframe 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 superframe. 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 superframe.
- 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 36-35 illustrates this timing issue. See also Section 36.4.1.2, “TDM-to-ATM.”

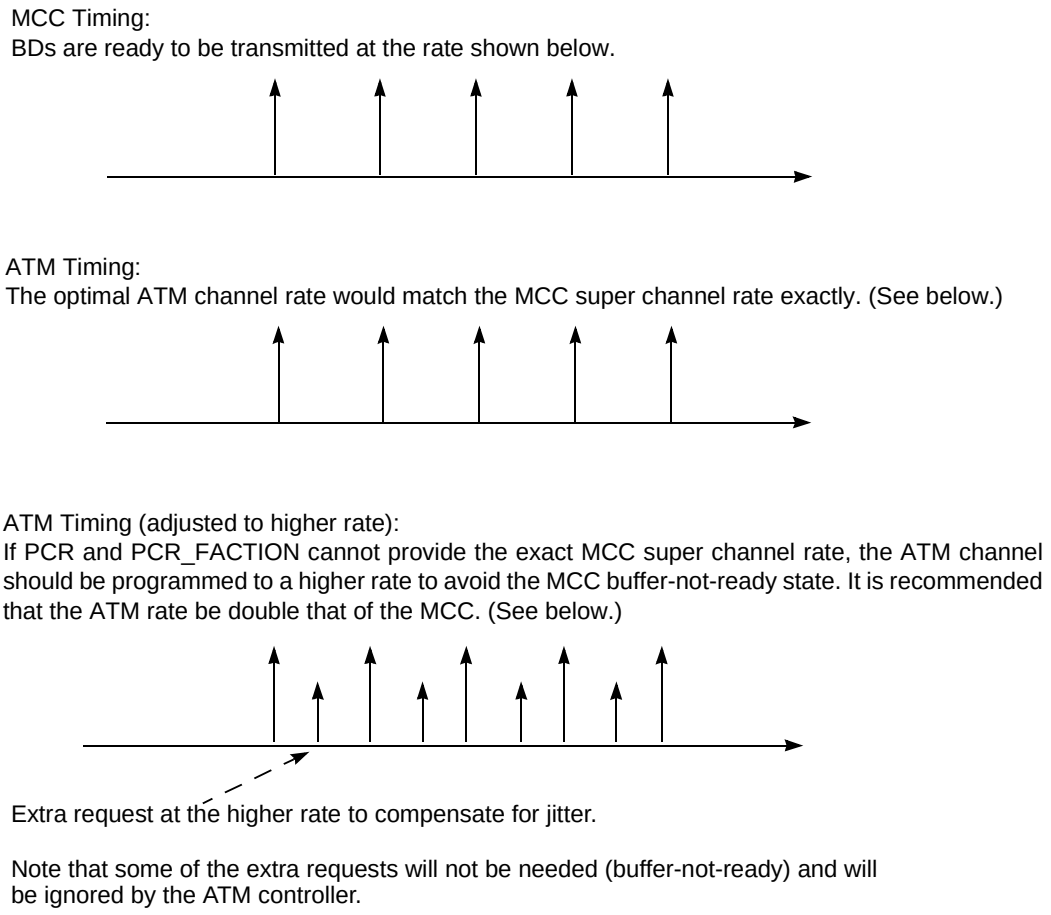


Figure 36-35. TDM-to-ATM Timing Issue

Chapter 37
AAL2

The 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 chapter describes the functionality and data structures of AAL2 CPS, CPS switching, and SSSAR and should be used as a supplement to 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.

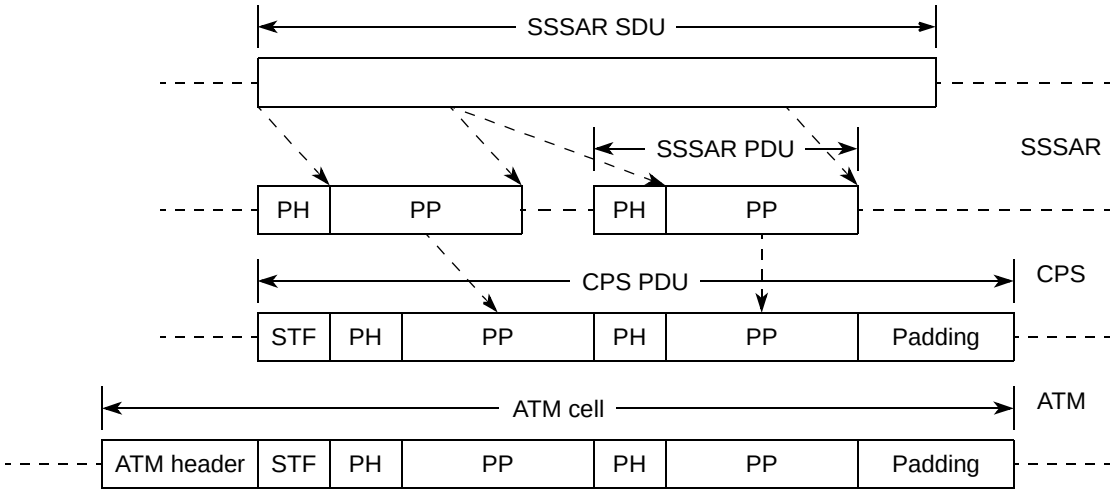
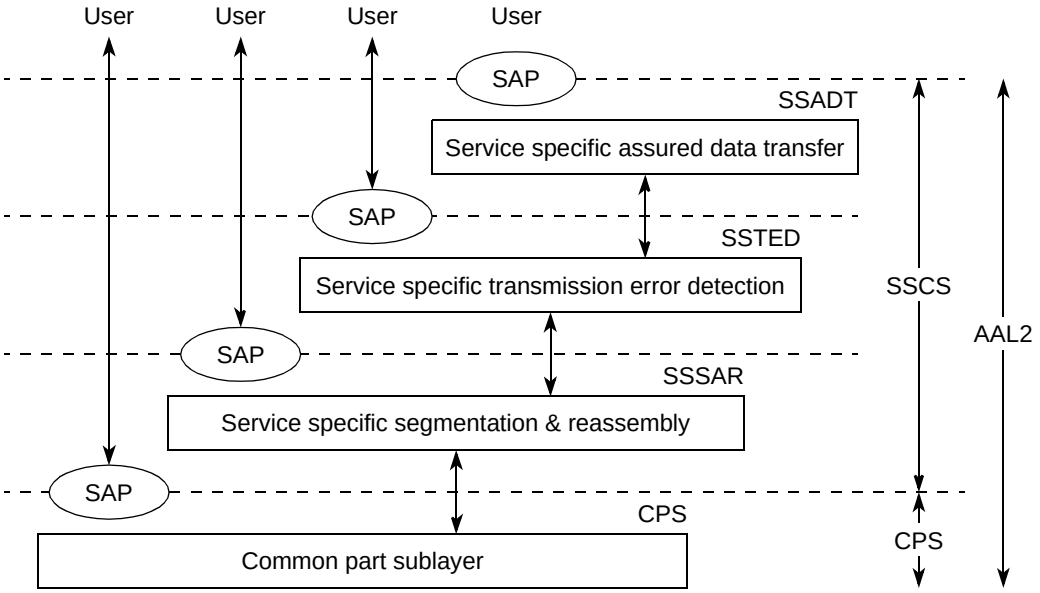


Figure 37-1. AAL2 Data Units

AAL2 is subdivided into two sublayers, as shown in Figure 37-2:

- Common part sublayer (CPS)—In the CPS sublayer, variable length packets coming from multiple users are assembled into CPS-PDUs belonging to a single ATM VC.
- Service-specific convergence sublayer (SSCS)—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:

— Service-specific segmentation and reassembly sublayer (SSSAR)
— Service-specific transmission error detection sublayer (SSTED)
— Service-specific assured data transfer sublayer (SSADT)



(SAP): Service access point

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.

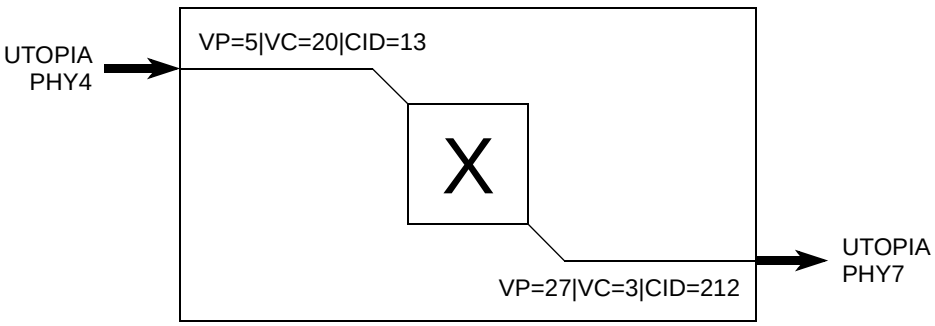


Figure 37-3. AAL2 Switching Example

37.2 Features

The MPC8560's AAL2 features are as follows:

- Fully complies with ITU-T I.363.2 (09/97 and 11/00) and ITU-T I.366.1 (06/98) specifications.
- Number of AAL2 external channels supported is subject to internal memory constraints
  - Each external channel requires space for one 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 Section 35.7, "User-Defined Cells (UDC)").
- 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 Protocol-Specific TCT”) 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 or taken from the buffer descriptor based on the user configuration. 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.)

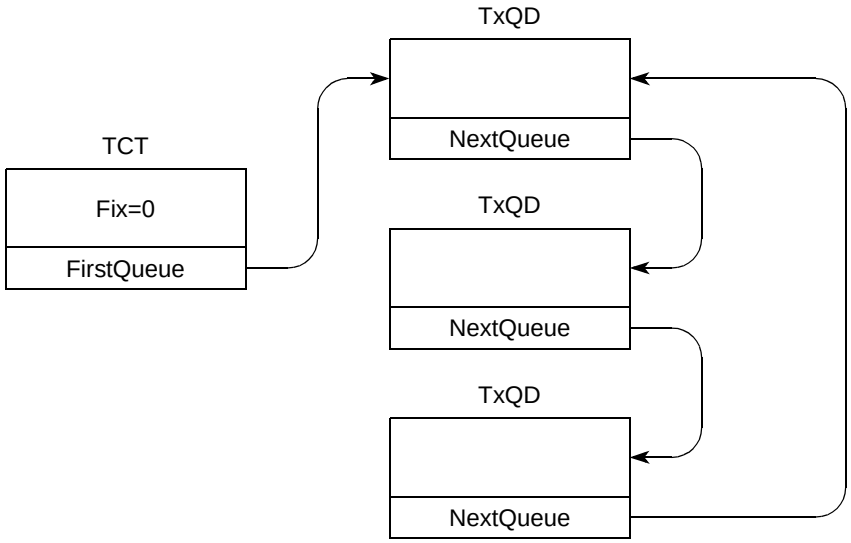


Figure 37-4. Round Robin Priority

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.

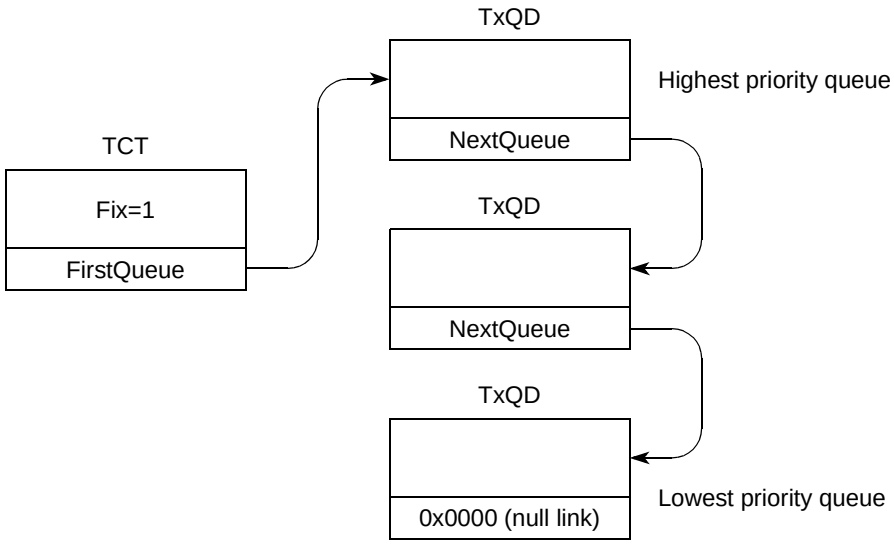


Figure 37-5. Fixed Priority Mode

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 TIMER CU mode as described above for the round robin mode.

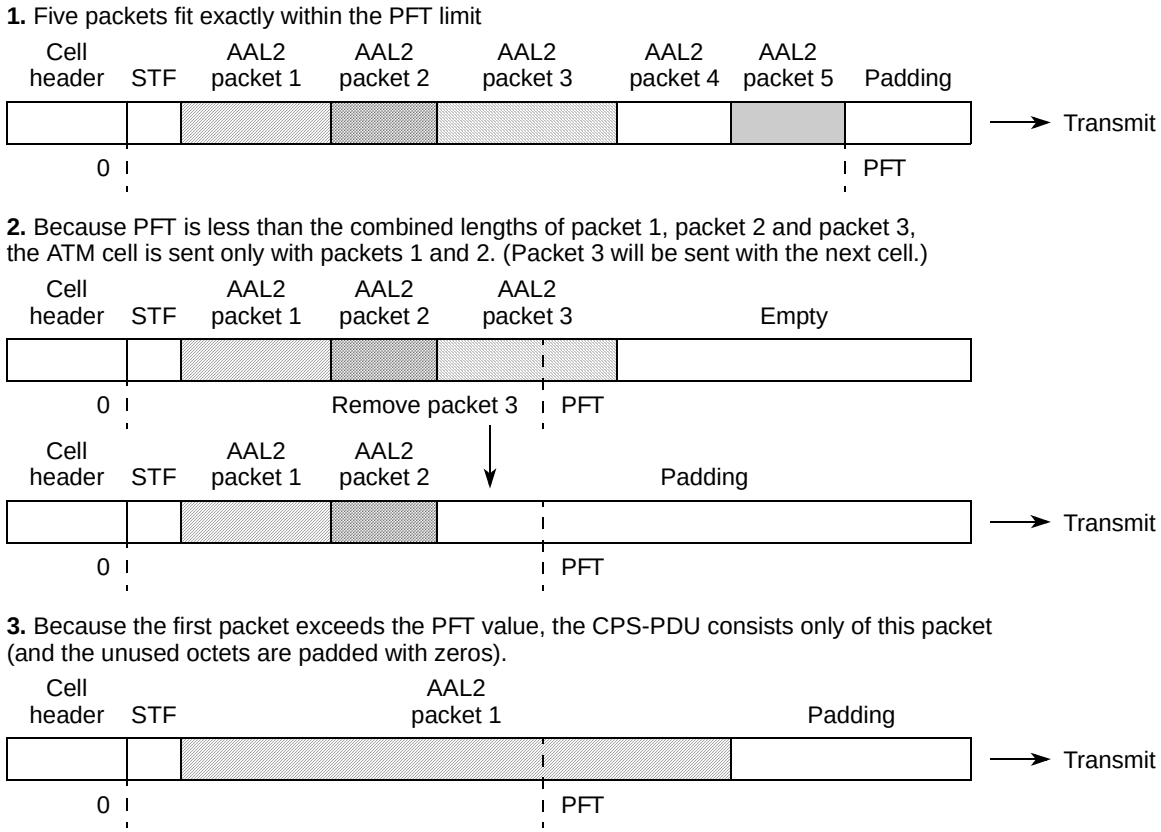
37.3.3 Partial Fill Mode (PFM)

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:



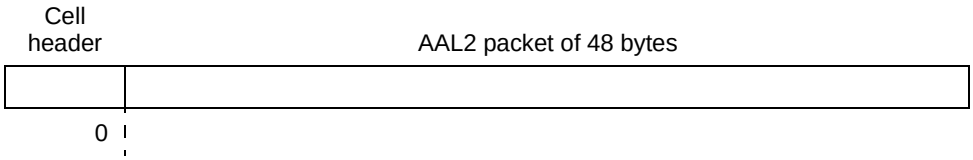
37.3.4 No STF Mode

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 doing the following:

1. Setting TCT[NoSTF]=1 and TCT[PFM]=1
2. Establishing a partial fill threshold by programming TCT[PFT]=47

The following figure shows a cell using no-STF mode:



37.3.5 AAL2 Tx Data Structures

The following sections describe the transmit connection tables (TCT) and the structures in which CPS packets and SSSAR SDUs are stored in memory.

37.3.5.1 AAL2 Protocol-Specific TCT

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 37-6 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 37-6. AAL2 Protocol-Specific Transmit Connection Table (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 Protocol-Specific Transmit Connection Table (TCT) Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–1	—	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. 00 Reserved. 01 Power PC little endian. 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 Reserved.
	7	**BIB**	Bus selection for the BDs, interrupt queues and the free buffer pool. 0 Reside on the system bus. 1 Reserved.
	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 Protocol-Specific Transmit Connection Table (TCT) Field Descriptions (continued)

Offset	Bits	Name[1]	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 010 AAL5 —ATM adaptation layer 5 protocol 100 AAL2 —ATM adaptation layer 2 protocol 101 AAL1_CES. All others reserved.
0x04	—	—	Reserved, should be cleared during initialization.
0x06	—	—	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	—	—	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 Protocol-Specific Transmit Connection Table (TCT) Field Descriptions (continued)

Offset	Bits	Name[1]	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.
	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.

1 **Boldfaced** entries 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 37-7. 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 37-7. CPS Tx Queue Descriptor (TxQD)

Table 37-2 describes the CPS TxQD fields.

Table 37-2. CPS TxQD Field Descriptions

Offset	Bits	Name[1]	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 descriptor.
	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 (continued)

Offset	Bits	Name[1]	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.

1 **Boldfaced** entries 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 contain the packet payload. See Figure 37-8.

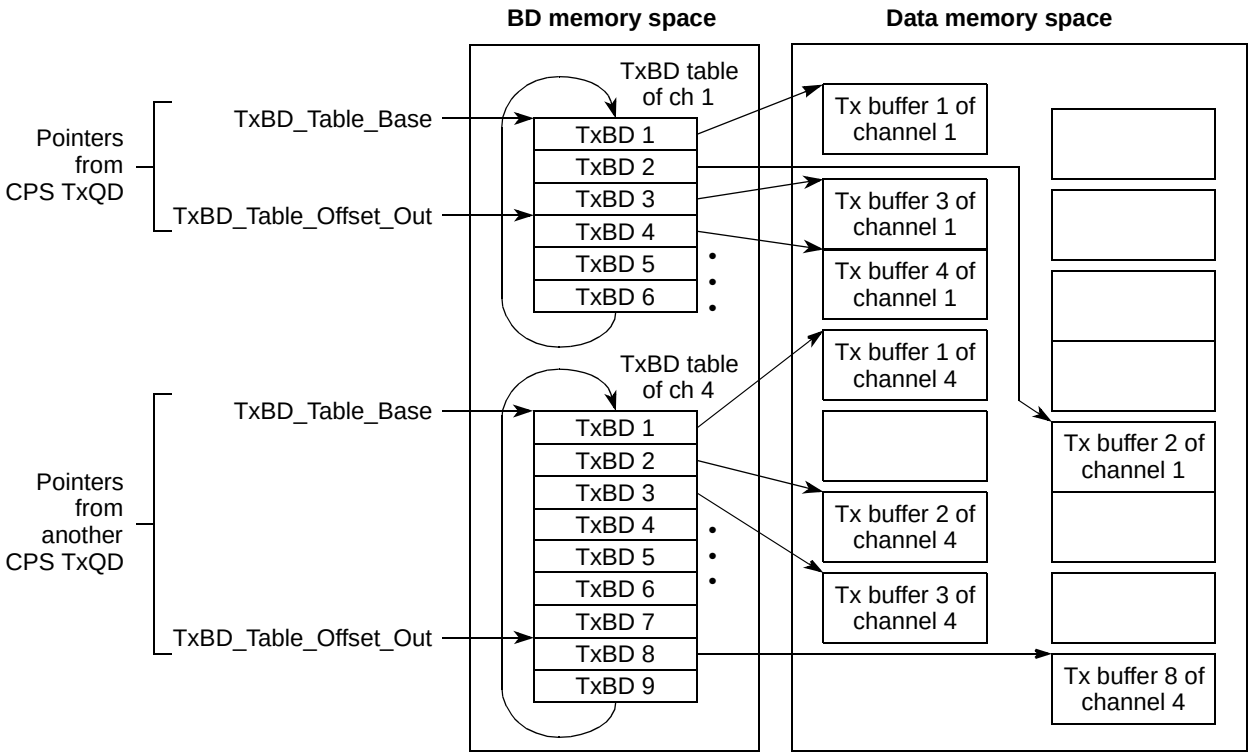


Figure 37-8. Buffer Structure Example for CPS Packets

Figure 37-9 shows a CPS TxBD.

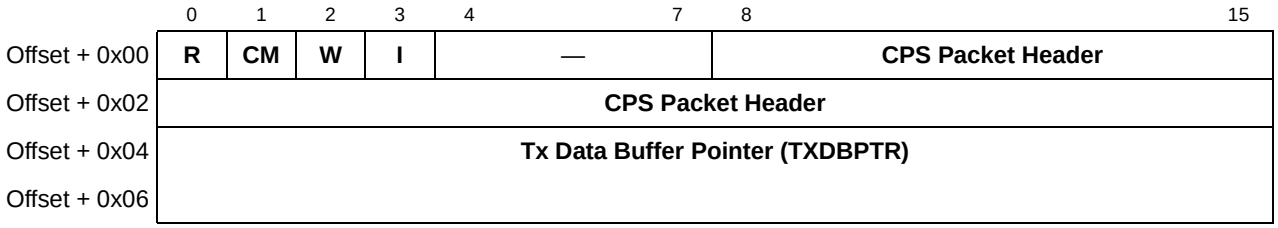


Figure 37-9. CPS TxBD

Table 37-3 describes the CPS TxBD fields.

Table 37-3. CPS TxBD Field Descriptions

Offset	Bits	Name[1]	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[2]**	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 **Boldfaced** entries must be initialized by the user.

2 **Setting Continuous mode (TxBD[CM] = 1) is not allowed in CID switching mode.**

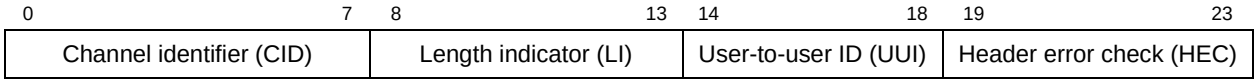


Figure 37-10. CPS Packet Header Format

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 37-11. 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.

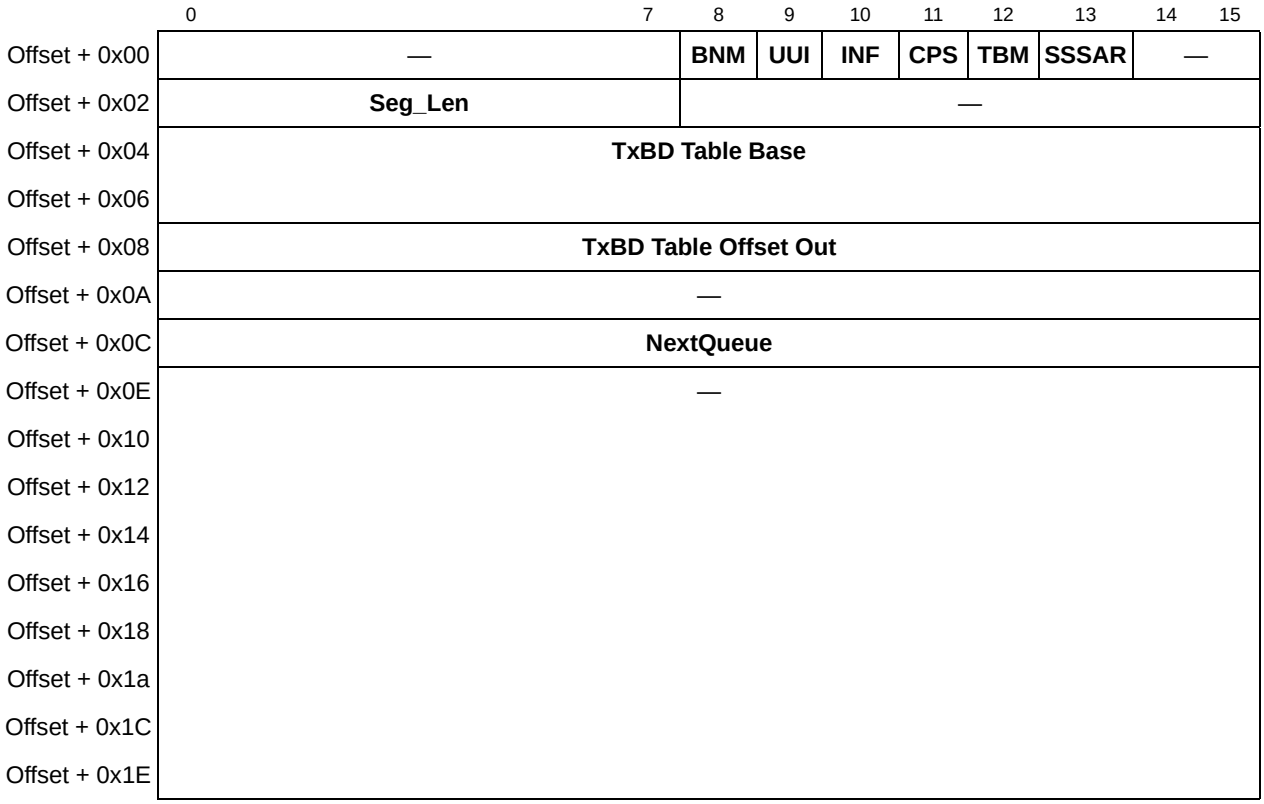


Figure 37-11. SSSAR Tx Queue Descriptor

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.
0x0A	—	—	Reserved, should be cleared during initialization.
0x0C	—	**NextQueue**	Points to the next TxQD to be serviced after this one. See Section Section 37.3.2, “Transmit Priority Mechanism.””
0x0E–0x1E	—	—	Reserved, should be cleared during initialization.

1 **Boldfaced** entries 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 37-12.

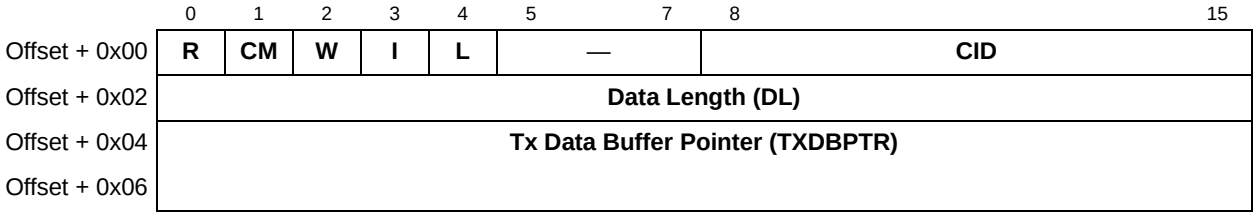


Figure 37-12. SSSAR TxBD

Table 37-5. SSSAR TxBD Field Descriptions

Offset	Bits	Name[1]	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[1]	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.

1 **Boldfaced** entries 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 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.

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 that have the same number but that are 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.

The mapping mechanism, shown in Figure 37-13, 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*4).
- Because the three types of RxQDs are different sizes, some offset numbers may not be used.

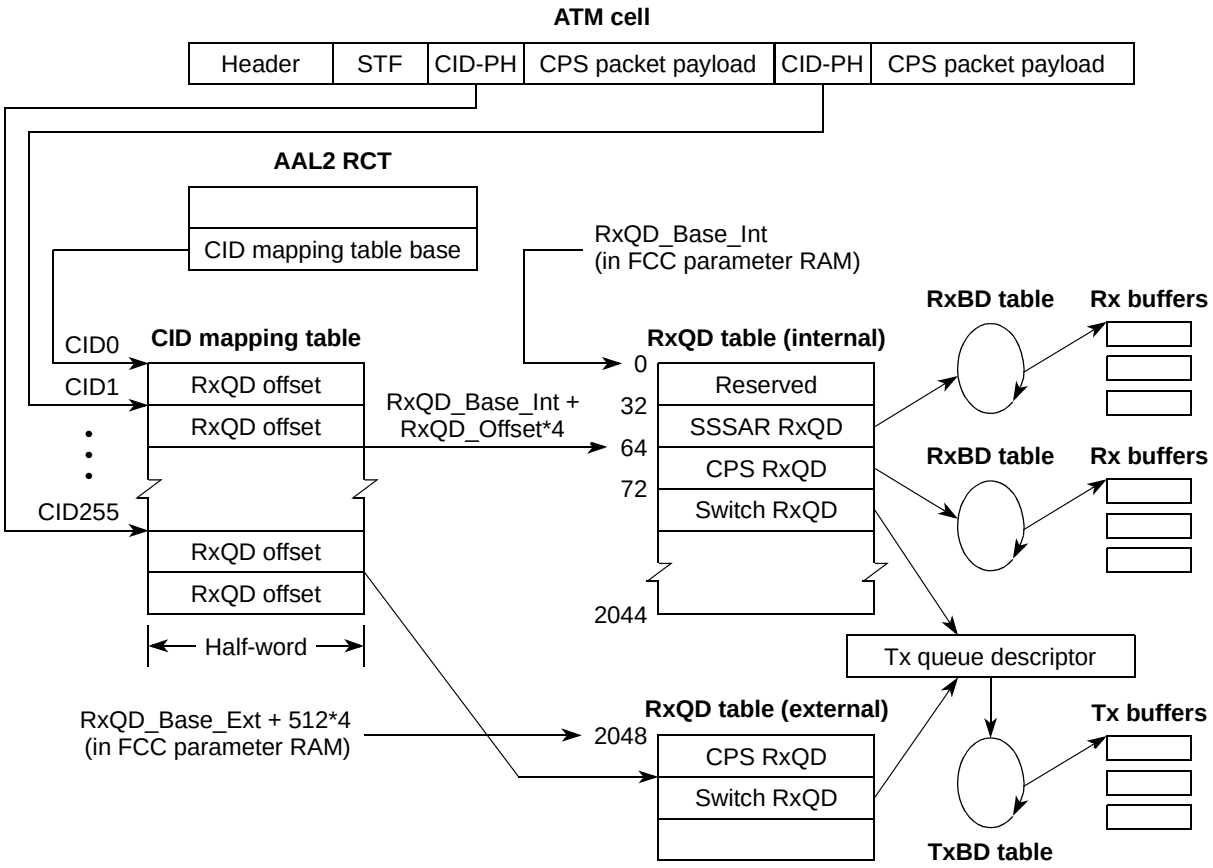


Figure 37-13. CID Mapping Process

37.4.3 AAL2 Switching

Switching is performed by pointing an RX CID at a switch RxQD (see Figure 37-14). 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.

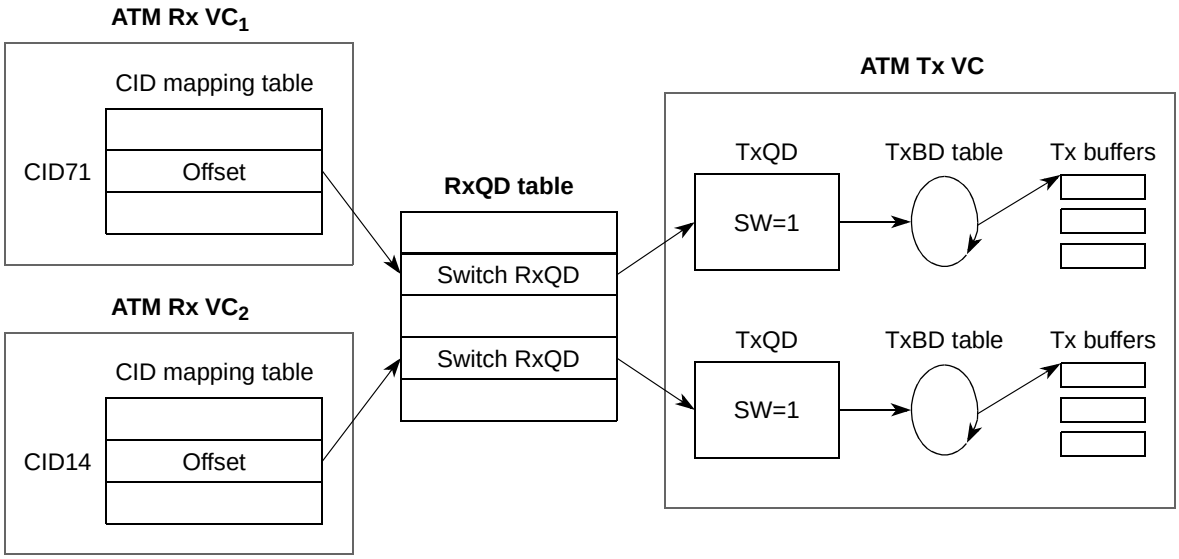


Figure 37-14. AAL2 Switching

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.

On any open BD that is partially filled, the receiver sets the UP (Un-complete packet) bit. When the packet is fully received during a normal error-free operation, the UP mark is removed, the Empty bit is set, and operation continues. However, if an error is detected by the receiver, the Empty bit is set and the UP bit remains set. In such a case, the transmitter skips this BD and proceeds to the next one. If for any reason the receiver that was in the middle of the BD stopped receiving traffic, the UP bit remains set and the Empty bit is cleared. Another receiver using the

same BD ring monitors the UP bit in addition to the Empty bit. If the UP bit is set, the other receiver does not proceed with the reception and gives a Busy interrupt. See Figure 37-17.

The receiver that is in a “stuck” state marks the BD with the receiver channel code that received the partial packet so that the host intervention is easier if needed. See Figure 37-19.

If the TBNR Time Out CNT mechanism is used, the transmitter advances after it tries to transmit the same BD with UP set after a given amount of attempts. The BD will be freed up for use. Refer to the TBNR Time Out CNT description in Table 37-6.

37.4.4 AAL2 RX Data Structures

The following sections describe the receive connection tables and the structures in which CPS packets and SSSAR SDUs are stored in memory.

37.4.4.1 AAL2 Protocol-Specific RCT

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.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0x00	—		GBL	BO		—	DTB	BIB	—		SEGF	ENDF	—	MAP	INTQ	
Offset + 0x02	—												NoSTF	AAL		
Offset + 0x04	—															
Offset + 0x06																
Offset + 0x08																
Offset + 0x0A																
Offset + 0x0c																
Offset + 0x0e																
Offset + 0x10																
Offset + 0x12																
Offset + 0x14																
Offset + 0x16																
Offset + 0x18	CID Mapping Table Base															
Offset + 0x1A																
Offset + 0x1C	—		PMT						TBNR Time OUT CNT							
Offset + 0x1E	Max_CPS_SDU_Deliver_length								—						EM	PM

Figure 37-15. AAL2 Protocol-Specific 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 Protocol-Specific RCT Field Descriptions

Offset	Bits	Name [1]	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. 00 Reserved 01 munged little endian 1x Big endian
	5	—	Reserved, should be cleared during initialization.
	6	**DTB**	Data buffer bus selection. 0 Reside on the system bus. 1 Reserved.
	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 Reserved.
	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	**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.

Table 37-6. AAL2 Protocol-Specific RCT Field Descriptions

Offset	Bits	Name [1]	Description
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 010 AAL5 —ATM adaptation layer 5 protocol 100 AAL2 —ATM adaptation layer 2 protocol 101 AAL1_CES. All others reserved.
0x04	—	—	Reserved, should be cleared during initialization.
0x06	—		
0x08	—		
0x0A	—		
0x0C	—		
0x0E	—		
0x10	—		
0x12	—		
0x14	—		
0x16	—		
0x18	—	**CID Mapping Table Base**	Points to the base address of the CID mapping table (see Figure 37-13). 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*32.
	8–15	**TBNR Time Out CNT**	The TBNR Time-Out CNT is a parameter that describes the amount of attempts the transmitter tries to transmit a packet on a BD ring which is current marked as partially filled, i.e. waiting for a receiver to finish reception of a packet. This value will be used internally by the transmitter that is the destination for this packet and decremented by the CPM on each attempt. Upon reaching the value 1, the transmitter will act as if the receiver is stuck in error condition and proceed to the next BD in the BD ring. This parameter is valid in switch mode only and should be programmed to a higher value than the ratio between the transmitter rate and the lowest receiver rate in the BD ring. The 8 bit value is scaled by 4 (setting TBNR Time-Out CNT =1 yields a value of 4 internally) so that the max number is 1K. Clearing this field will disable this feature completely

Table 37-6. AAL2 Protocol-Specific RCT Field Descriptions

Offset	Bits	Name [1]	Description
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.

1 **Boldfaced** entries must be initialized by the user.

37.4.4.2 CID Mapping Tables and 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].

There are two RxQDs—one for internal RxQDs and one for external RxQDs. 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*RxQD_Offset.

Offsets between 512–65535 belong to the external RxQD table. The address of an external RxQD is RxQD_Base_Ext + 4*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

Each CPS RxQD, as shown in Figure 37-16, points to an CPS RxBD table.

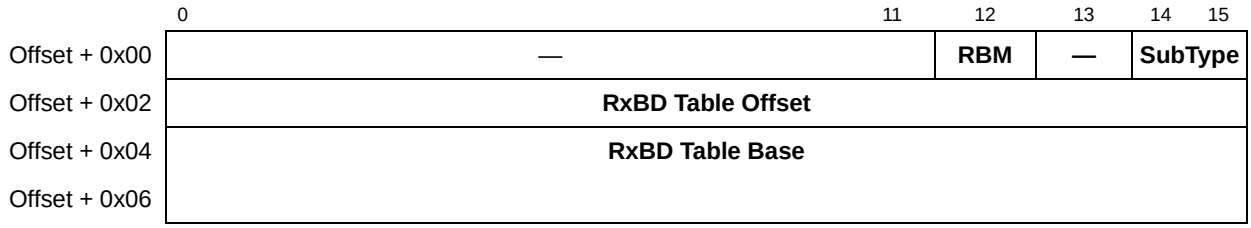


Figure 37-16. CPS Rx Queue Descriptor

Table 37-7 describes the CPS RxQD fields.

Table 37-7. CPS RxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–11	—	Reserved, should be cleared during initialization.
	12	**RBM**	Receive buffer mask. 0 Disable receive buffer interrupt. 1 Enable receive buffer interrupt.
	13	—	Reserved, should be cleared during initialization.
	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.

1 **Boldfaced** entries must be initialized by the user.

37.4.4.4 CPS Receive Buffer Descriptor (RxBD)

The CPS RxBD 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 37-17. The buffers contain the packet payload.

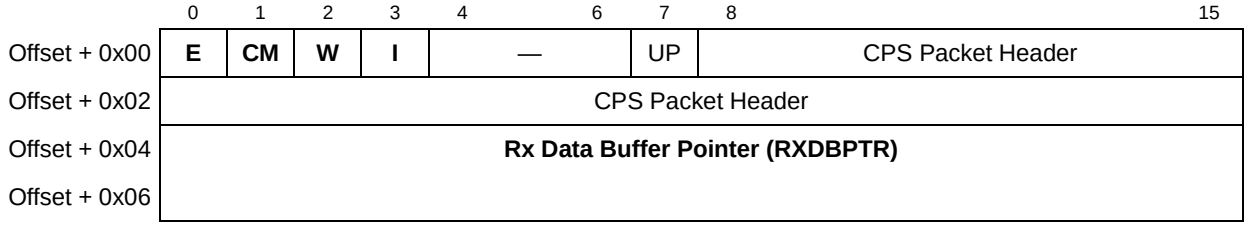


Figure 37-17. CPS Receive Buffer Descriptor

Table 37-8 describes the CPS RxBD fields.

Table 37-8. CPS RxBD Field Descriptions

Offset	Bits	Name[1]	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, should be cleared during initialization.
	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.

1 **Boldfaced** entries must be initialized by the user.

37.4.4.5 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 37-9 describes the CPS switch RxQD fields.

Table 37-9. CPS Switch RxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0-7	**TX CID**	Translation CID. The received CID is saved in a TX Queue with this new CID number.
	8-11	—	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.

1 **Boldfaced** entries must be initialized by the user.

37.4.4.6 SWITCH Receive/Transmit Buffer Descriptor (RxBD)

The switch 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. This BD is common to the receiver and the transmitter.

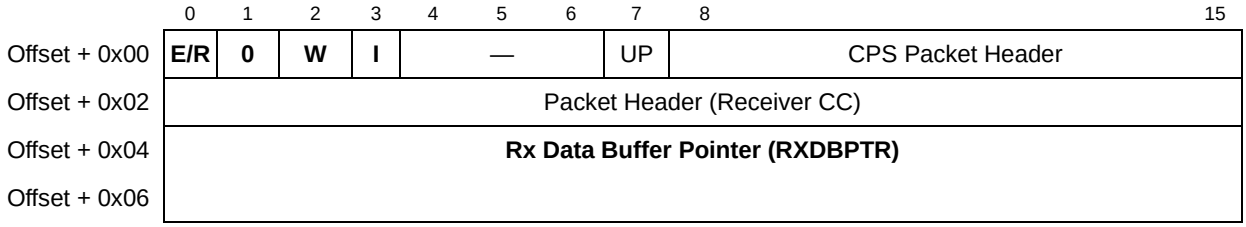


Figure 37-19. Switch Receive/Transmit Buffer Descriptor

Table 3-11 describes the Switch RxBD fields.

Table 37-10. Switch RxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0	**E/R**	Buffer Ready Must be set to zero.
	1	**0**	Not valid for switching mode, should be cleared to 0 upon 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 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, should be cleared during initialization.
	7	UP	Uncompleted packet. 0 No error occurred in this packet and the complete packet has been received. 1 If R/E=1 a receive error occurred that caused this packet to be uncompleted. The receive error type is reported to the interrupt queue. The transmitter will skip this BD when in this state and continue to the next BD in the ring. If R/E=0 a receiver has received the first part of a packet and is waiting for the rest of it to be received on the next ATM cell.
	8–15	CPS Packet Header	Contains the beginning of the packet header. See Figure 37-10 for the CPS packet header format. (see remark in next row)

Table 37-10. Switch RxBD Field Descriptions

Offset	Bits	Name[1]	Description
0x02	—	CPS Packet Header (Receiver CC)	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. In case of a “stuck” receiver in switch mode, where the BD ring in common to Tx and Rx, this field indicates the last Receiver Channel Code number which has been received. The terminology for “stuck” implies a receiver which started receiving a packet and the rest of the packet hasn’t been received.When the receiver is in a “stuck” state the entry: CPS Packet Header is not valid. If the Time-out mechanism is being used this field is being used internally by the CPM.
0x04	—	**RXBDPTR**	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 **Boldfaced** entries must be initialized by the user.

37.4.4.7 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.

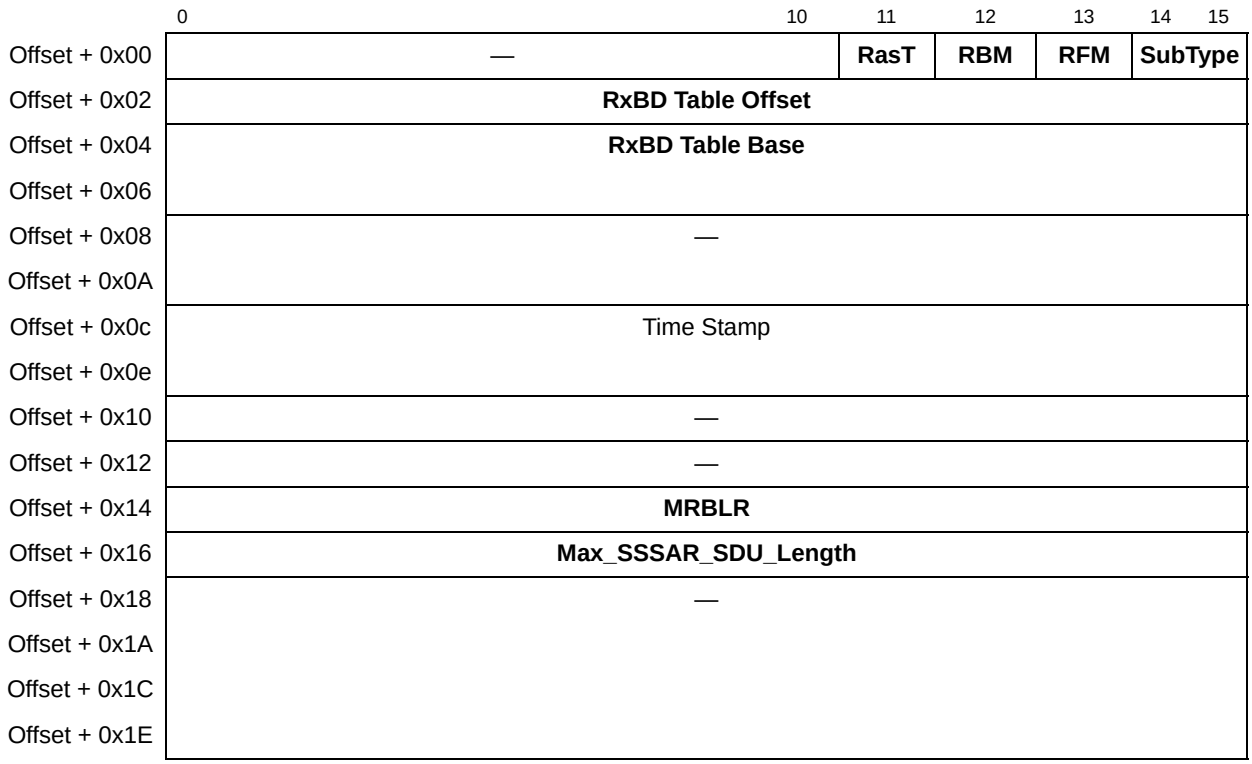


Figure 37-20. SSSAR Rx Queue Descriptor

Table 37-11 describes the SSSAR RxQD fields.

Table 37-11. SSSAR RxQD Field Descriptions

Offset	Bits	Name[1]	Description
0x00	0–10	—	Reserved, should be cleared during initialization.
	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
0x02	—	**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.
0x0A	—	—	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).
0x10	—	—	Reserved, should be cleared during initialization.
0x12	—	—	Reserved, should be cleared during initialization.
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.
0x18–0x1E	—	—	Reserved, should be cleared during initialization.

1 **Boldfaced** entries must be initialized by the user.

37.4.4.8 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.

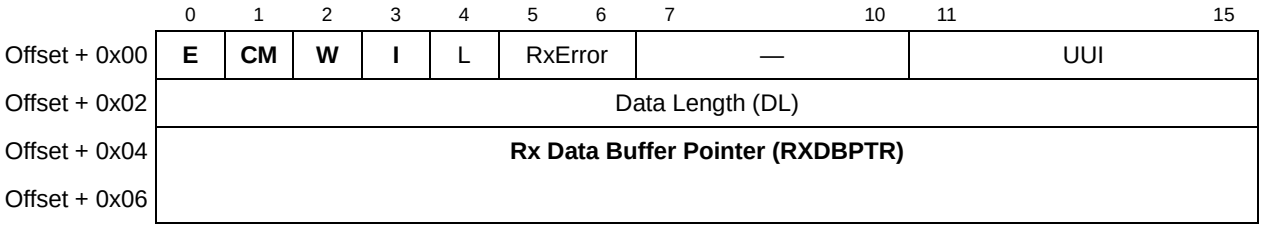


Figure 37-21. SSSAR Receive Buffer Descriptor

Table 37-12 describes the SSSAR RxBD fields.

Table 37-12. SSSAR RxBD Field Descriptions

Offset	Bits	Name[1]	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.[2] 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 **Boldfaced** entries must be initialized by the user.

2 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.

37.5 AAL2 Parameter RAM

When configured for ATM mode, the FCC parameter RAM is mapped as shown in Table 37-13. 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 37-13. 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.

Table 37-13. AAL2 Parameter RAM (continued)

Offset	Name	Width	Description
0x66	INTT_BASE	Hword	Interrupt queue parameters table base. User-defined.
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).”
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 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)”).
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 37-13. 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*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*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*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*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.
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 as described in Section 35.7, “User-Defined Cells (UDC),” also applies to AAL2 operation. 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 37-13), 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# *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#*16 (where CH# is the ATM channel number)

The structure of a UDC header table (receive or transmit) is shown in Figure 37-22.

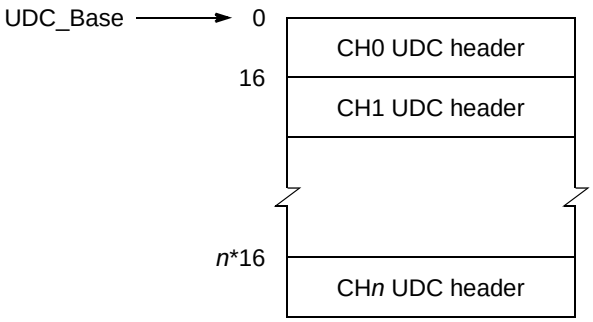


Figure 37-22. UDC Header Table

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.

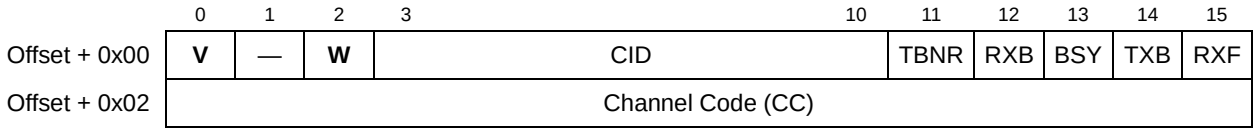


Figure 37-23. AAL2 Interrupt Queue Entry CID ≠ 0

Table 37-14 describes the interrupt queue entry fields for a CID.

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 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. **Note:** The CID number that is placed in the interrupt queue is the one currently located in the last BD. Because the CID is not updated when the BD is not ready, the CID value is the one extracted from this BD when it was last processed and transmitted. If the BD is never processed and the BD was cleared, the CID value could be zero.
	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 toTable 35-41 for more information.

An interrupt entry for the VC is shown in Figure 37-24.

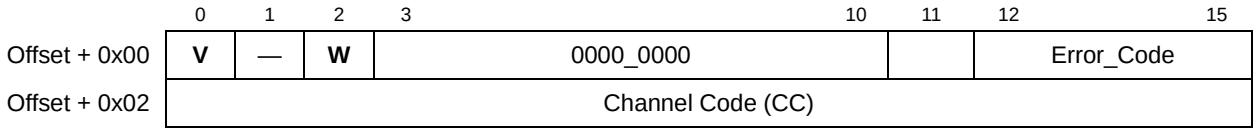


Figure 37-24. AAL2 Interrupt Queue Entry CID = 0

Table 37-15 describes the interrupt queue entry fields for the VC. All the receive error events are enabled by setting RCT[EM].

Table 37-15. 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 through 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.

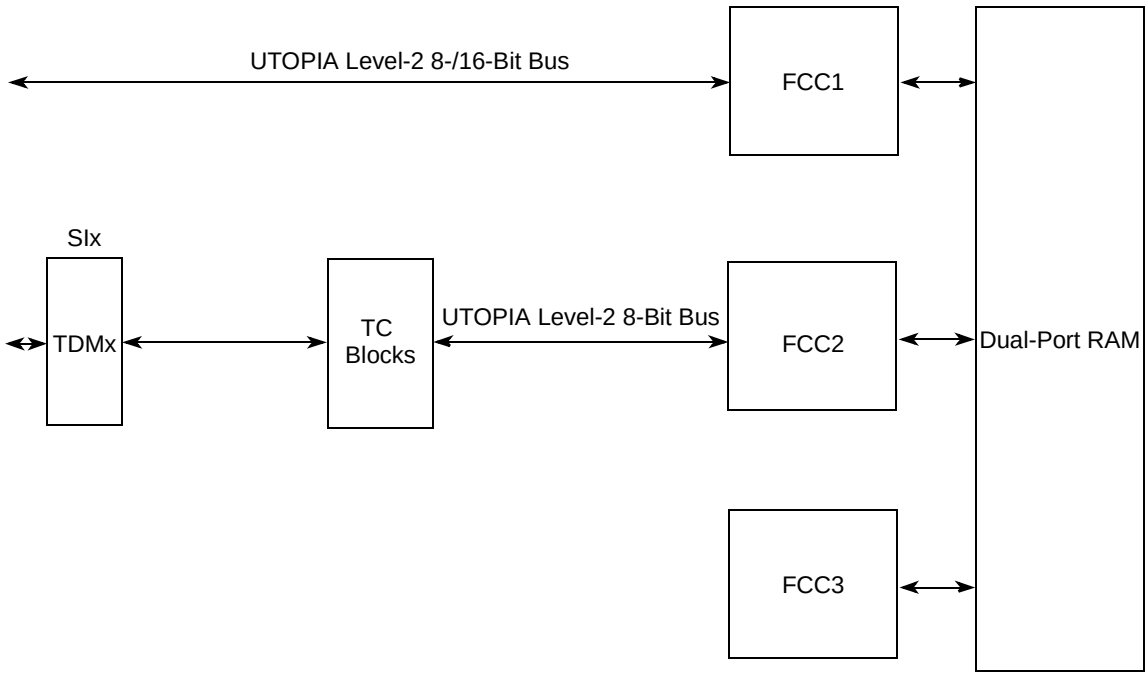


Figure 38-1. Serial ATM Using FCC2 and TC Blocks (Single Channel)

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 errored cells
    - – HEC single bit errored 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.

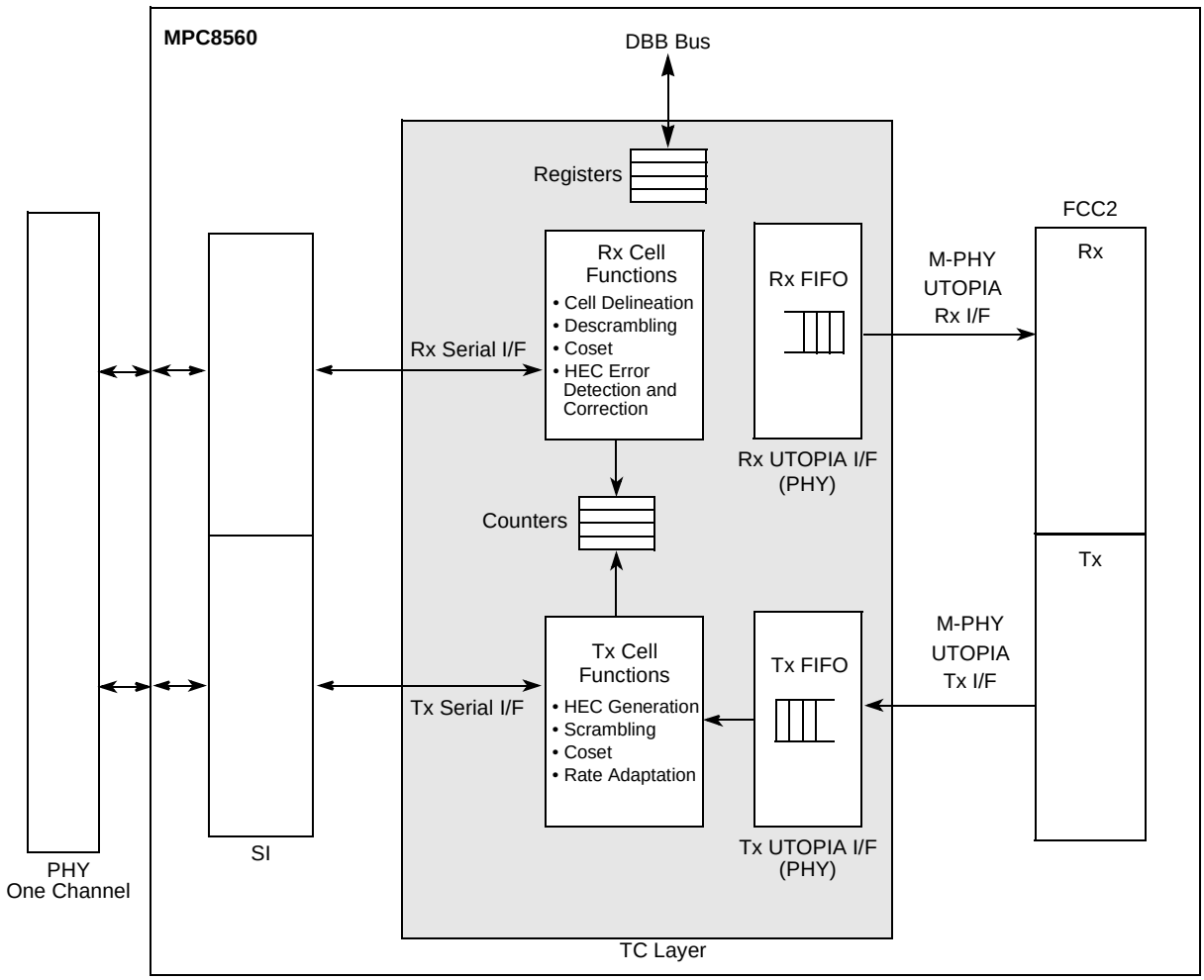


Figure 38-2. TC Layer Block Diagram

38.3 Signals

The TC layer is operated through 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.

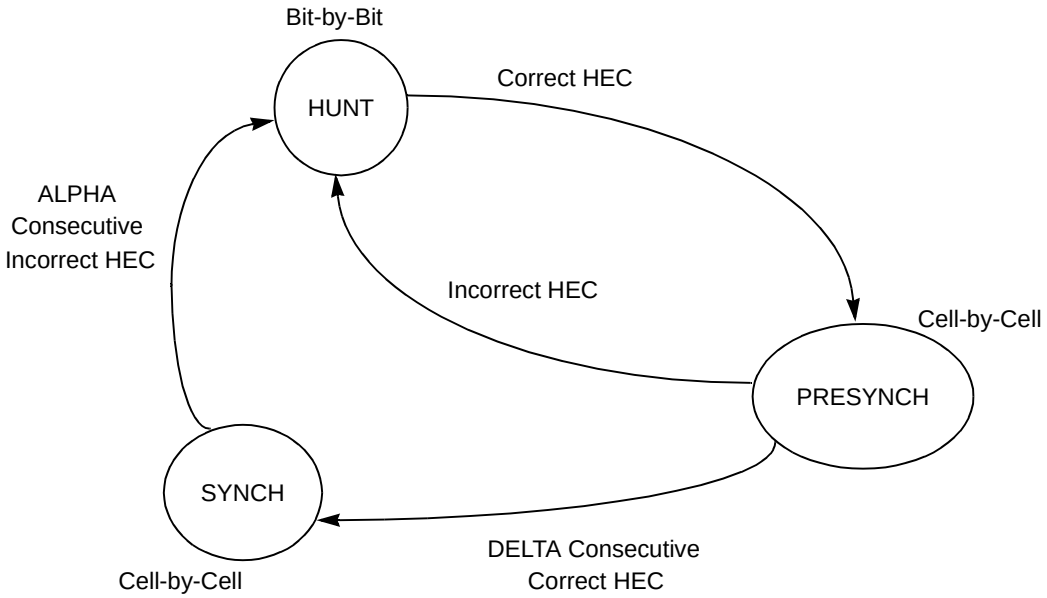


Figure 38-3. TC Cell Delineation State Machine

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 errored headers under bursty error conditions.

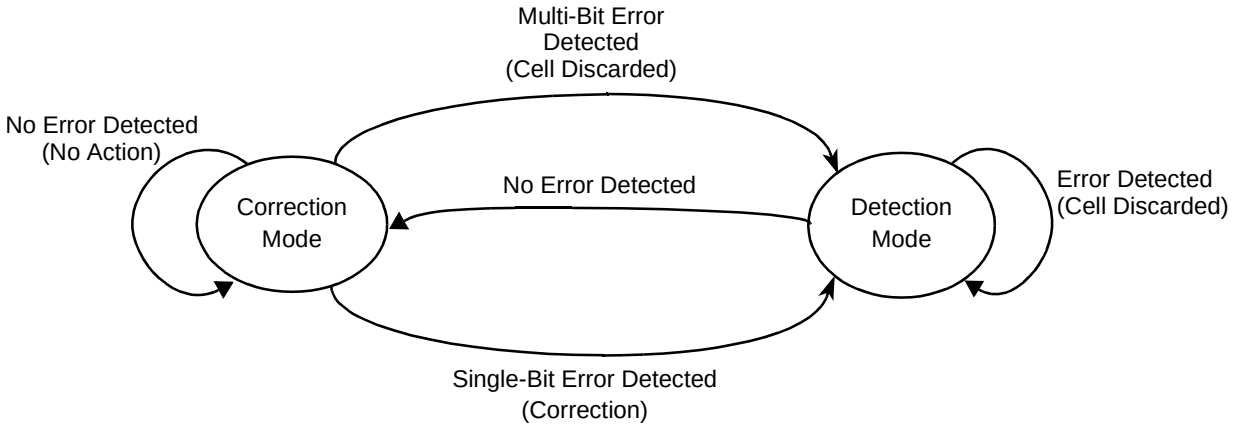


Figure 38-4. HEC: Receiver Modes of Operation

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 through 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 through 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	11 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															
Addr	TCMODE1: 0x9_1400, TCMODE2: 0x9_1420, TCMODE3: 0x9_1440, TCMODE4: 0x9_1460, TCMODE5: 0x9_1480, TCMODE6: 0x9_14A0, TCMODE7: 0x9_14C0, TCMODE8: 0x9_14E0															

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

Table 38-3. TCMODE Field Descriptions (continued)

Bits	Name	Description
2	RPS	Rx Payload DeScrambling 0 Payload descrambling is performed on received payload data. 1 No payload descrambling is performed on received payload data.
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
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 (CDSMR*x*)

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															
Addr	CDSMR1: 0x9_1402, CDSMR2: 0x9_1422, CDSMR3: 0x9_1442, CDSMR4: 0x9_1462, CDSMR5: 0x9_1482, CDSMR6: 0x9_14A2, CDSMR7: 0x9_14C2, CDSMR8: 0x9_14E2															

Figure 38-7. CDSMR*x*

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 (TCER*x*)

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															
Addr	TCER1: 0x9_1404, TCER2: 0x9_1424, TCER3: 0x9_1444, TCER4: 0x9_1464, TCER5: 0x9_1484, TCER6: 0x9_14A4, TCER7: 0x9_14C4, TCER8: 0x9_14E4															

Figure 38-8. TCER*x*

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	Errored cells counter overflow Set when the errored 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 (TCMR*x*)

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															
Addr	0x9_1502															

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															
Addr	0x9_1500															

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 Errored Cell Counter (ECC)

This cell counter is updated whenever a received errored 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 errored cells are counted by the errored 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.

38.11.5 Programming 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 through 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 through 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 (through 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 through the UTOPIA interface. When a cell is passed through 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 through 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.

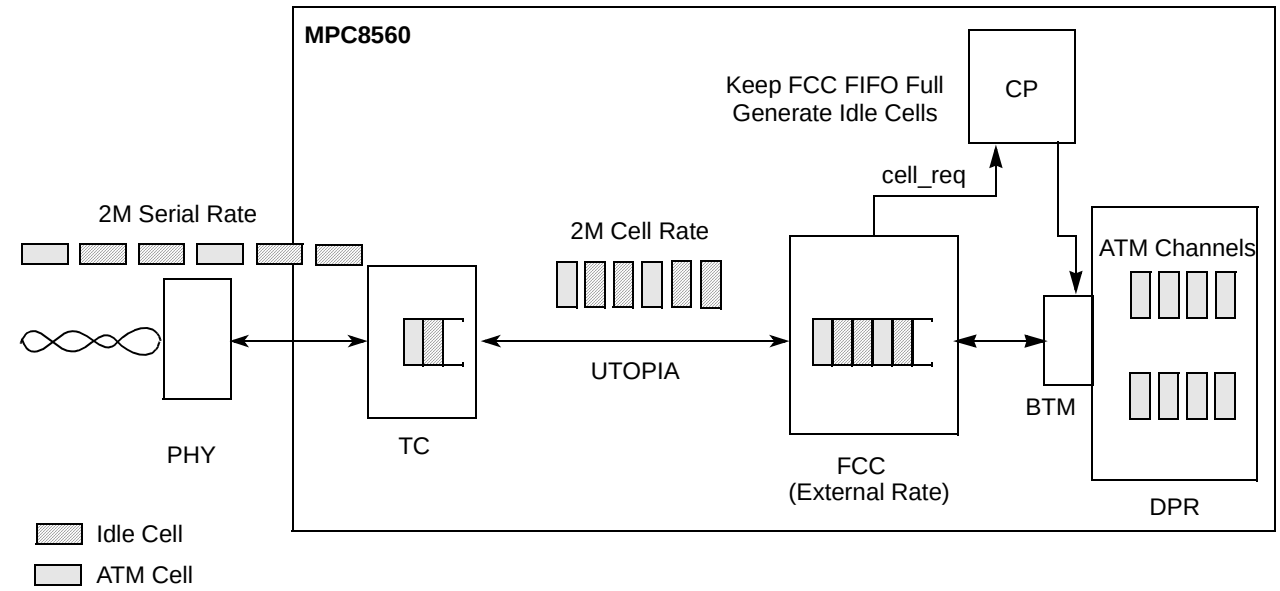


Figure 38-12. TC Operation in FCC External Rate Mode

- 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 through the UTOPIA bus from the FCC. See Figure 38-13.

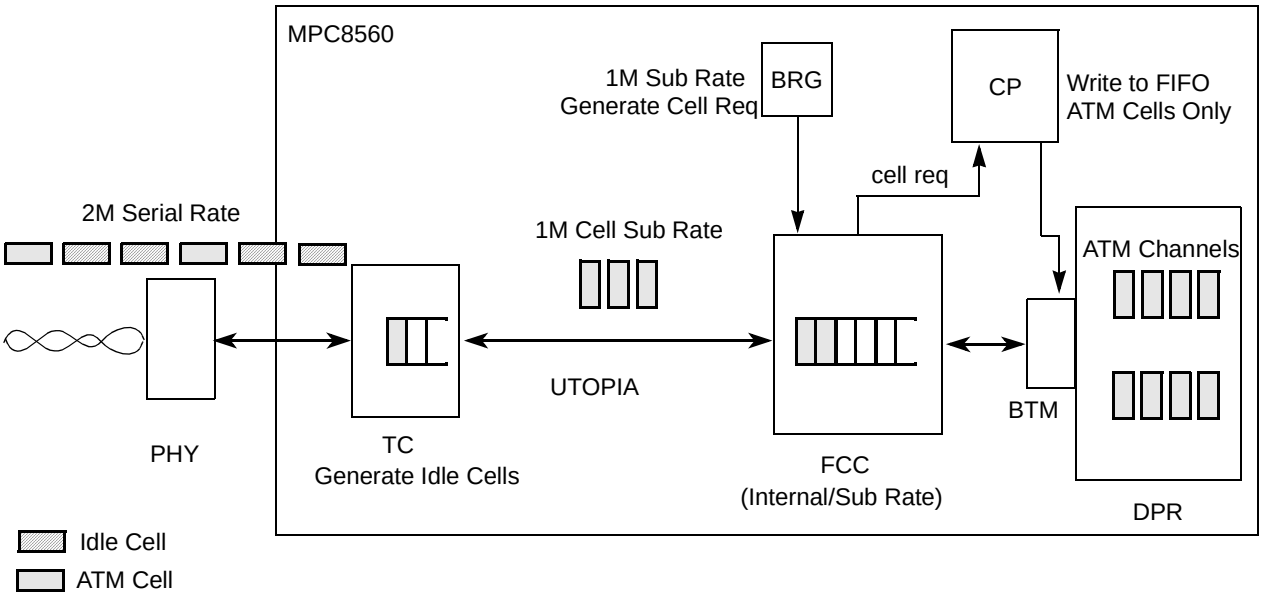


Figure 38-13. TC Operation in FCC Internal Rate Mode (Sub Rate Mode)

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 through the SI to the eight TC layer blocks. The eight TC layer blocks, each with its own address, are connected internally to FCC2 through the UTOPIA 8-bit bus. Another ATM stream is managed by FCC1 through the UTOPIA 16-bit bus connected to a SONET 155-Mbps PHY.

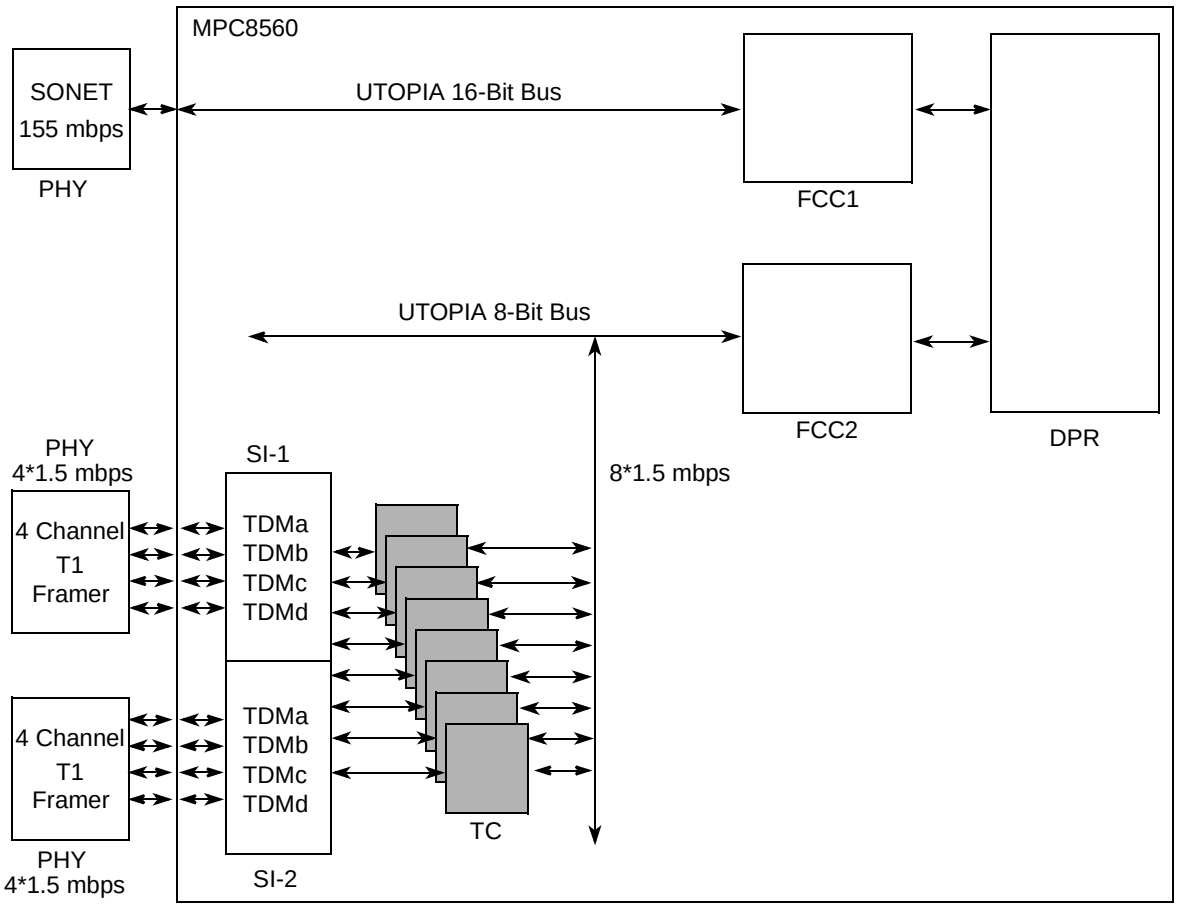


Figure 38-14. Example of Serial ATM Application

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
Inverse Multiplexing for ATM (IMA)

The MPC8560 provides advanced ATM functionality through its implementation of inverse multiplexing for ATM (IMA). Unlike many of its predecessor PowerQUICC II devices that provide IMA capability in microcode, the MPC8560's IMA functionality is ROM-resident. This chapter provides a broad overview of the IMA specifications and the specific implementation available on the MPC8560.

39.1 Features

The IMA ATM Forum Specification defines the functions shown in Table 39-1.

Table 39-1. IMA Sublayer in Layer Reference Model

Layer	Sublayer	User Plane Functions	Layer Management Functions	Plane Management Functions
ATM	—	—	—	—
Physical	IMA specific transmission convergence	• ATM cell stream splitting and reconstruction • ICP cell insertion and removal • Cell rate decoupling • IMA frame synch • Stuffing • Discarding bad-HEC cells	• IMA connectivity • ICP cell errors (OIF) • LIF/LODS/RDI-IMA defect processing • RDI-IMA alarm generation • Tx/Rx IMA link state report	• IMA group configuration • Link addition/removal • ATM cell rate change • IMA group failure notification • IMA statistics
	Interface specific transmission convergence	• No cell discarding • No cell rate decoupling	—	—
		• Cell delineation • Cell scrambling and descrambling • Header error correction • HEC generation and verification	• HEC error indication • LCD-RDI alarm generation	• LCD failure notification • TC statistics
	Physical medium dependent	• Bit timing • Line coding • Physical medium	• Local alarm processing • RDI alarm generation	• Link failure notification • PMD state

The MPC8560's IMA implementation does not encompass all of these functions. Application software is responsible for managing the start-up procedure, handling changes in group control, status, and maintaining the Link State Machine and Group State Machine. In general, the

MPC8560 implements most of the IMA sublayer user plane functions and provides interrupts/statistics for the layer and plane management functions.

The key features of the MPC8560's IMA implementation are:

- Supports any FCC with multi-PHY UTOPIA capability
- Supports both IMA links and non-IMA links on the same FCC (on a per-PHY basis)
- Supports up to 8 IMA groups with one FCC
- Supports up to 8 links per IMA group using internal TC layer hardware
- Supports up to 31 links per IMA group using an external TC layer device (Note that a maximum of 31 total links can be supported either on FCC1 or FCC2 but not concurrently on both FCCs)
- Performs the following IMA User Plane functions
  - ATM cell stream splitting and reconstruction
  - ICP cell insertion/removal--communication of control and framing information
  - Cell rate decoupling--insertion of 'filler' cells when ATM layer cells are unavailable
  - IMA frame synchronization--finding IMA frame boundaries within links
  - Stuffing--insertion of 'stuff' cells into fast links, in order to maintain an average data rate between links of a group
  - Discards cells with bad HECs
- Delay synchronization--correlating IMA frames among links of a group
- Maximum differential delay supported is user-programmable
- Provides interrupts on errors/state changes
- Maintains low-level statistic counters
  - Transmit stuff events
  - Receive stuff events
  - Receive ICP violations
  - Receive Out-of-IMA Frame anomalies

39.1.1 References

The MPC8560's IMA features are driven by The ATM Forum's IMA specifications (available at www.atmforum.com):

- The ATM Forum Technical Committee. Inverse Multiplexing for ATM (IMA) Specification Version 1.1 - AF-PHY-0086.001
- The ATM Forum Technical Committee. Inverse Multiplexing for ATM (IMA) Specification Version 1.0 - AF-PHY-0086.000

39.1.2 IMA Versions Supported

The MPC8560's IMA implementation supports IMA version 1.0 and version 1.1, with only minor configuration changes. These include the following:

- Programming the IMA version encoding in the OAM labels of the transmit ICP and filler cell templates.
- Programming the validated OAM label field in the IMA group receive parameters.

39.1.3 PHY-Layer Devices Supported

The MPC8560's IMA implementation is primarily targeted at ATM's primary application (that is, IMA over multiple DS1/E1). However, it supports any UTOPIA PHY device which (1) has a constant data rate and (2) can be programmed not to screen out HEC-errored cells. Most PHYs have this mode available for IMA also.

39.1.4 ATM Features Not Supported

The following ATM features are not available for IMA links, but are available for non-IMA links:

- User-defined cells (UDC) (that is, cells that are not 53 bytes and/or with customized headers)
- Internal rate mode for APC scheduling

39.2 IMA Protocol Overview

This section describes the IMA protocol, not its specific implementation in the MPC8560.

39.2.1 Introduction

Inverse Multiplexing over ATM (IMA) provides a cost effective solution to carry high speed connections, for example, T3/E3 and OC-3c/STM-1 links, over already installed low speed connections, for example, T1/E1 links, in a flexible way by dynamically adding/removing links as required, depending on the bandwidth required.

IMA is defined as transmitting a stream of data over multiple low speed links and recombining the stream in the correct order at the end. IMA involves inverse multiplexing and de-multiplexing of ATM cells in a cyclical fashion among links grouped to form a higher bandwidth logical link whose rate is approximately the sum of the link rates. This is referred to as an IMA group. Figure 39-1 provides a simple illustration of ATM Inverse Multiplexing technique in one direction. This technique also applies in the opposite direction.

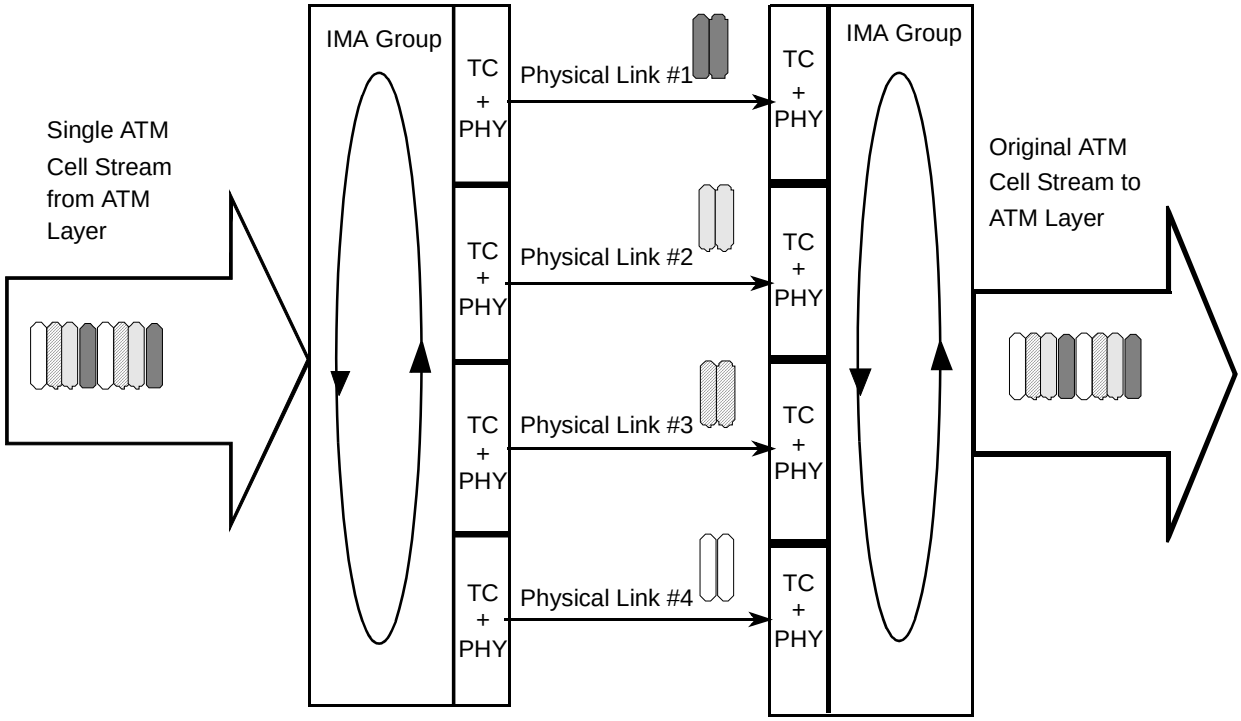


Figure 39-1. Basic Concept of IMA

In the transmit direction (near-end), the ATM cell stream received from the ATM layer is distributed on a cell by cell basis, across the multiple links within the IMA group. At the far-end, the receiving IMA unit recombines the cells from each link, on a cell by cell basis, recreating the original ATM cell stream. The aggregate cell stream is then passed to the ATM layer.The IMA protocol enables the demultiplexing/deconstruction (transmit) of an ATM cell stream into multiple links. When receiving, the IMA protocol multiplexes/reconstructs incoming cells from multiple links into the original ATM cell stream. The IMA protocol must compensate for differences in clock rate and delay over the multiple links.

39.2.2 IMA Frame Overview

The IMA interface periodically transmits special cells that contain information needed for reconstruction of the ATM cell stream at the receiving end of the IMA virtual link. The receiver end reconstructs the ATM cell stream after accounting for the link differential delays, smoothing CDV introduced by the control cells, etc. These cells, defined as IMA Control Protocol (ICP) cells, provide the definition of an IMA frame. The transmitter must align the transmission of IMA frames on all links (shown in Figure 39-2). This allows the receiver to adjust for differential link delays among the constituent physical links. Based on this required behavior, the receiver can detect the defensible delays by measuring the arrival times of the IMA frames on each link.

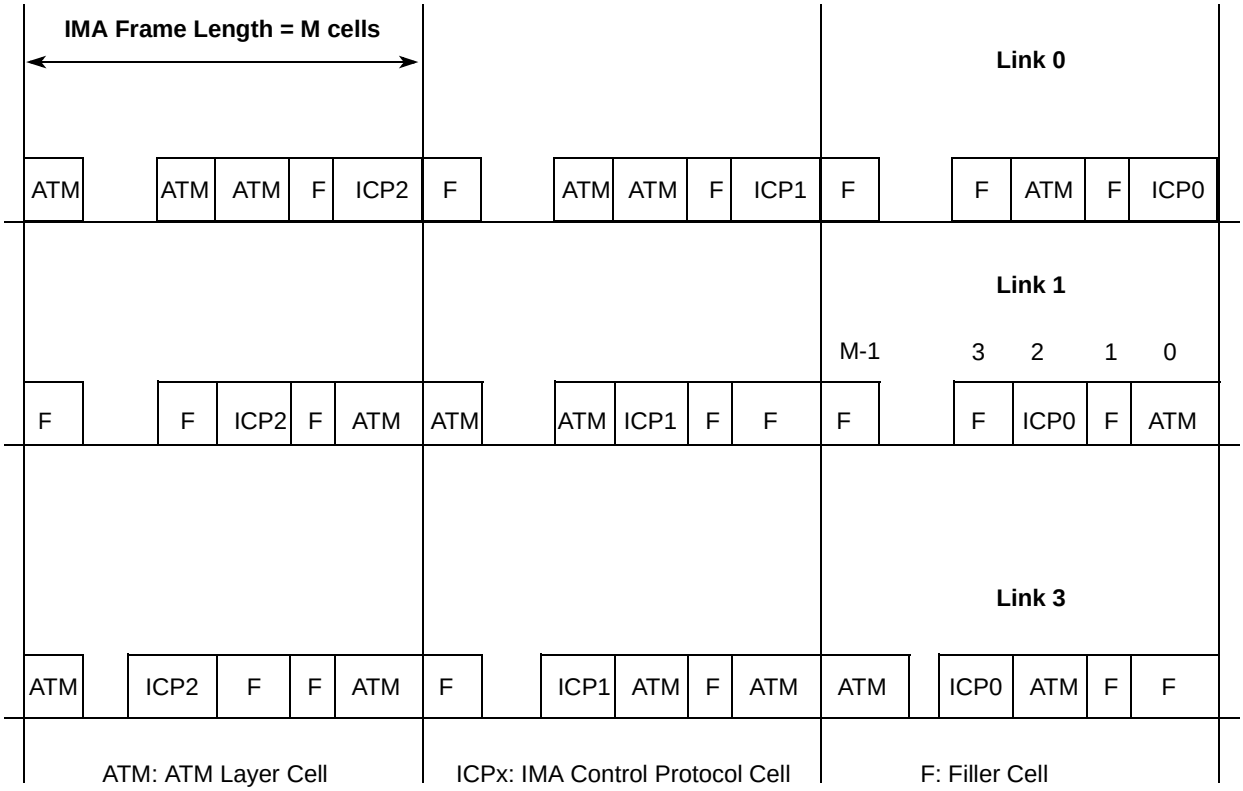


Figure 39-2. Illustration of IMA Frames

At the transmitting end, the cells are transmitted continuously. If there are no ATM layer cells to be sent between ICP cells within an IMA frame, then the IMA transmitter sends filler cells to maintain a continuous stream of cells at the physical layer. The insertion of filler cells provides cell rate decoupling at the IMA sublayer. The filler cells should be discarded by the IMA receiver.

A new OAM cell is defined for use by the IMA protocol. The cell has codes that define it as an ICP or filler cell.

The data multiplexing performed by IMA is cell-based, where cells are distributed among the links in the IMA group in a round-robin cycle. In order to compensate for different clock rates, IMA must periodically insert ‘stuff’ cells into faster links in order to maintain a consistent average data rate over the links of the group. Furthermore, IMA must compensate for potential differences in delay between the links of the group. Per the IMA specification, the allowable delay differential for DS1/E1 links is 25ms, which at E1 rates is equivalent to approximately 118 cells. The MPC8560 allows the user to define the allowable delay differential via a delay compensation buffer of programmable length.

IMA accomplishes these goals by the periodic insertion of special OAM cells, which (among other things) define M-cell frame boundaries and provide frame sequence numbers and stuffing information. This framing information is used by the receiver to correlate the received cell streams

and extract cells in-order from the links of the IMA group, thereby reconstructing the original cell stream.

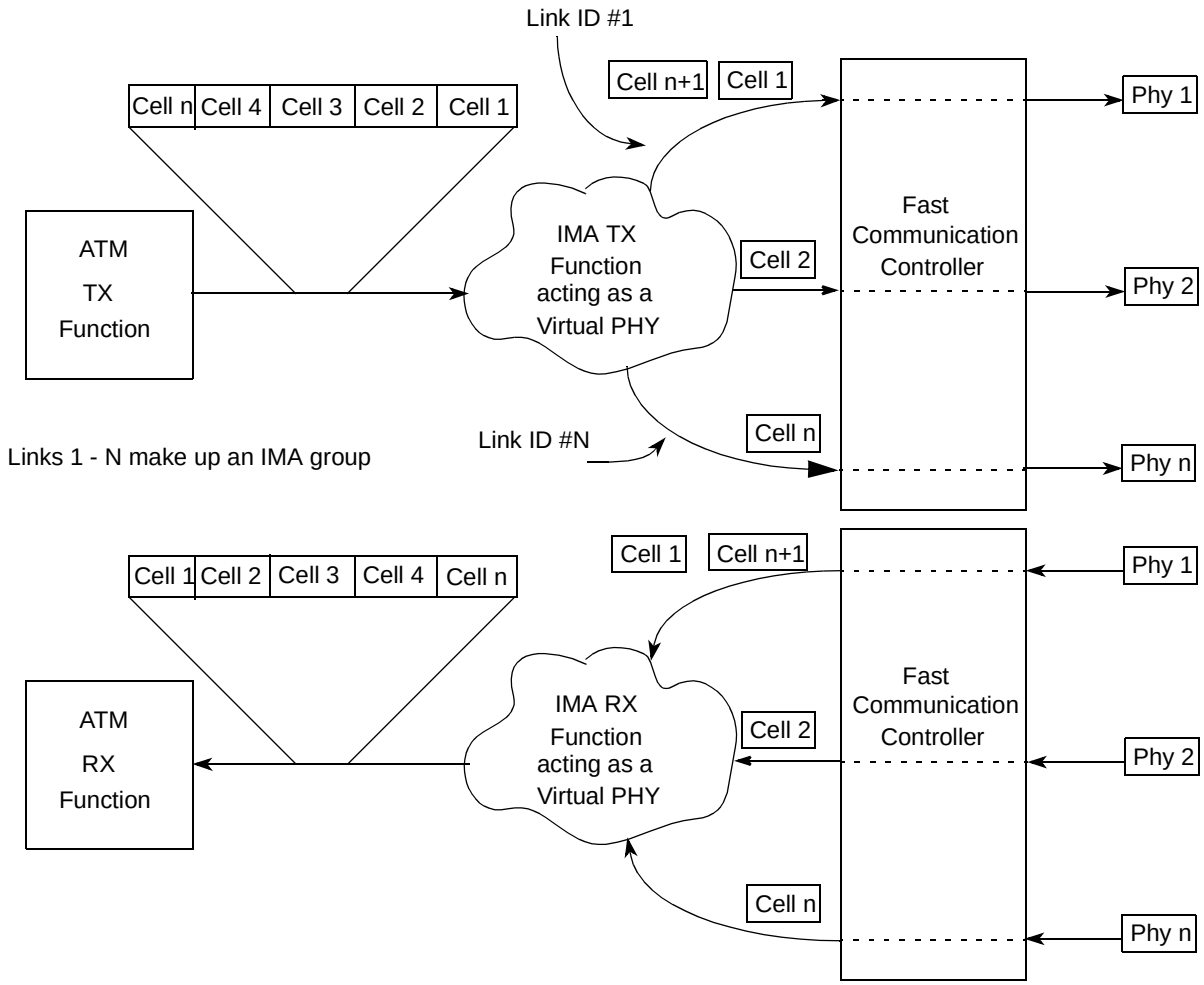


Figure 39-3. IMA Overview

39.2.3 Overview of IMA Cells

An IMA frame consists of M number of cells (M = 32, 64, 128, or 256 cells). Each frame consists of ATM data cells and IMA control cells. Two types of IMA control cells are defined by and used by the IMA protocol; IMA Control Protocol (ICP) cells and filler cells.

39.2.3.1 IMA Control Cells

There is at least one control cell (ICP) in each frame. An additional ICP cell may be included in a frame to compensate for timing differences between the links in an IMA group (for example, if one link is slightly faster than the other). The insertion of additional ICP cells to compensate for timing differences between links is called a “stuff event”. The transmitter is responsible for

inserting “stuff” ICP (SICP) cells and the receiver will monitor for stuff indication and discard SICP cells. The location of the ICP cell in an IMA frame is determined during the IMA start-up sequence.

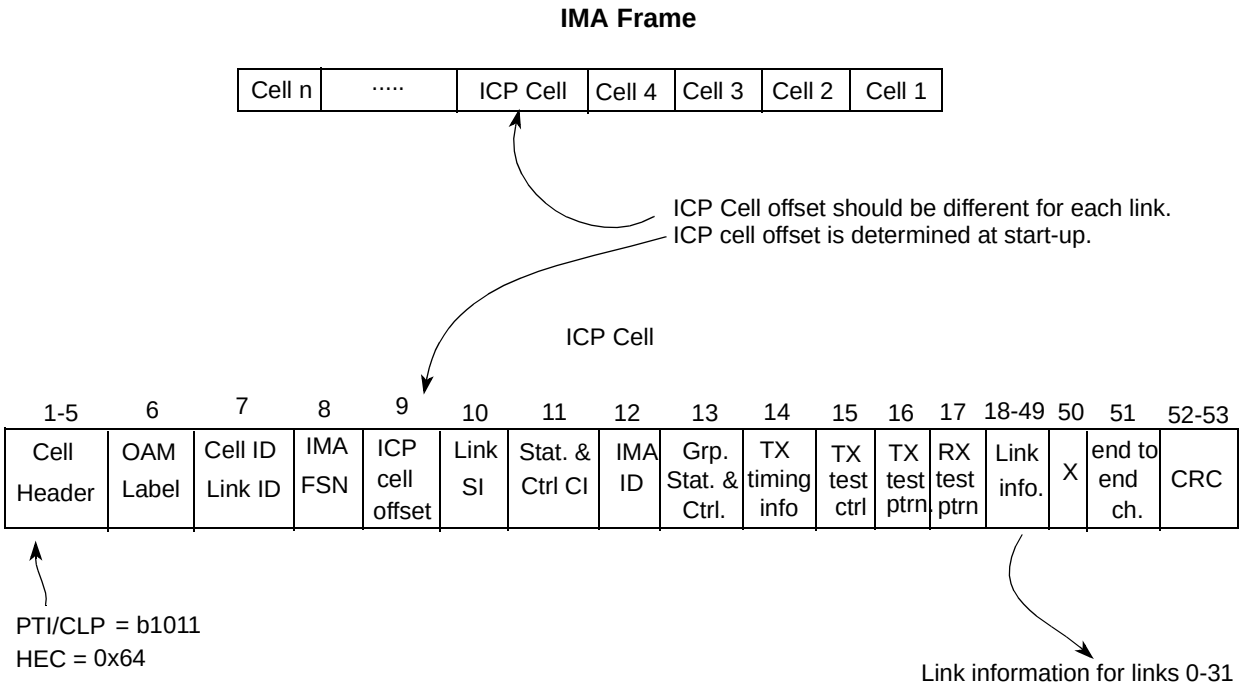


Figure 39-4. IMA Frame and ICP Cell Formats

The IMA protocol must compensate for potential differences in delay between the different links of the IMA group. The allowable delay differential for DS1/E1 links is 25ms, which at E1 rates is equivalent to approximately 118 cells.

39.2.3.2 IMA Filler Cells

At the transmitting end, cells are transmitted continuously. If there are no ATM layer cells to be sent between ICP cells within an IMA frame, then the IMA transmitter sends filler cells to maintain a continuous stream of cells at the physical layer. The insertion of filler cells provides cell rate decoupling at the IMA sublayer. The filler cells should be discarded by the IMA receiver.

39.3 IMA Implementation Architecture

This section explains the architecture of the receive and transmit IMA implementation tasks.

39.3.1 IMA Function Partitioning

The MPC8560 IMA implementation performs only those functions with regular, critical real-time demands. The other functions of IMA (for example, control and management) are the

responsibility of host software. As such, the MPC8560 IMA implementation corresponds primarily to the user plane functions defined in the IMA specification, and software must provide the layer management and plane management functionality. The MPC8560 provides interrupts when interaction with layer management and plane management is required.

39.3.1.1 User Plane Functions Implemented

- ATM cell stream splitting and reconstruction
- ICP cell insertion/removal
- Cell rate decoupling (that is, filler cell insertion/removal)
- IMA frame synchronization
- Stuffing
- Discards cells with bad HECs (available on 0.25µm (HiP4) Rev. B silicon and forward)

39.3.1.2 Plane Management Functions Implemented

As stated above, most plane management functions must be performed in software. However, certain statistics are intimately related to the lower-level user plane functions, and are provided. These include the following:

- ICP violations
- Transmit stuff events
- Receive stuff events

39.3.2 Transmit Architecture

This section discusses the behavior of the MPC8560's IMA functionality during transmission, focusing particularly on the independent transmit clock (ITC) mode of IMA. Differences in behavior when common transmit clock (CTC) mode is used are discussed at the end of this section.

Only one cell scheduler (known as the ATM pace controller or APC) is used per IMA group. This APC schedules transmission for the IMA group as a single aggregate channel. The APC hands these cells off to the MPC8560 IMA Tx, which distributes these scheduled cells to each of the PHYs in the IMA group. To compensate for clocking differences such as jitter and average speed differential, the IMA Tx process distributes ATM cells into N jitter buffers with a depth of five cells. The IMA PHYs take cells from these jitter buffers and transmit them.

The cell scheduling is triggered by requests from the timing reference link (that is, assertion of TxClav from the TRL's PHY). Requests from non-TRL PHYs only interact with the jitter buffer, and perform the stuffing function as needed, when the jitter buffer becomes too shallow. Therefore, the tasks performed in response to TRL PHY requests and non-TRL PHY requests are different.

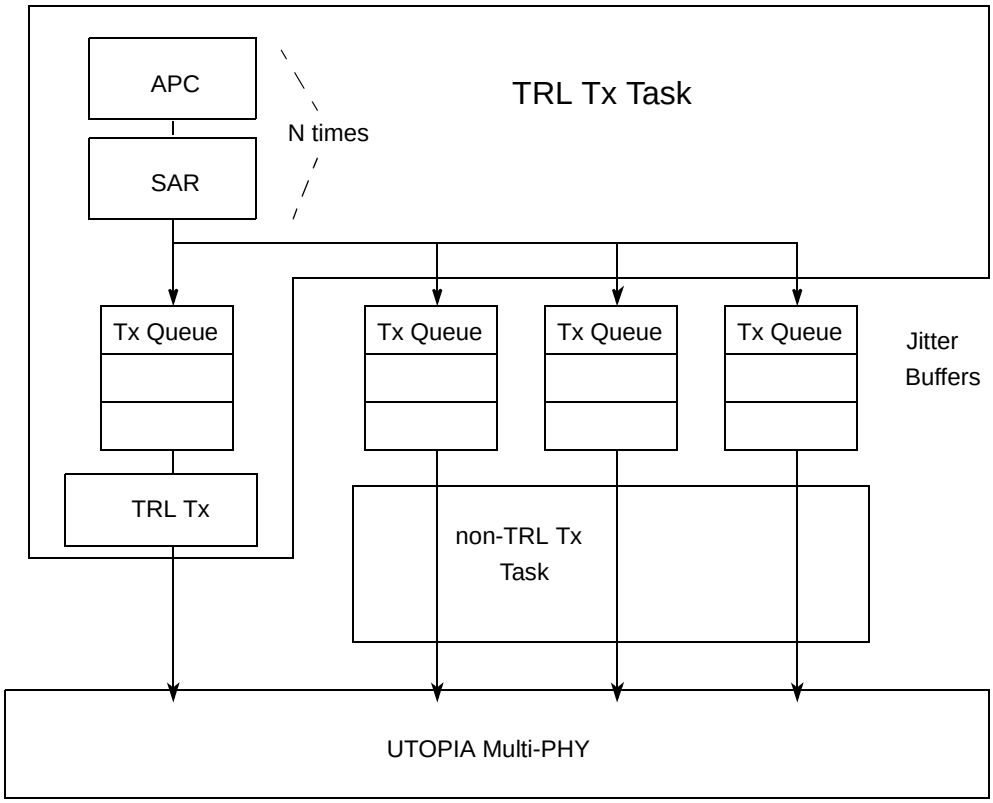


Figure 39-5. IMA Transmit Task Interaction

39.3.2.1 TRL Operation

A request from the TRL PHY is used to trigger a complete round-robin process of cell scheduling and distribution, distributing one cell for each of the transmit queues of the N links in the IMA group. For each link, the MPC8560 will:

1. Determine whether an ICP cell or a data/filler cell should be sent for the link:
   — If data/filler, determine whether link is in ‘active’ or ‘filler-only’ mode
   — If active, run the APC scheduling algorithm to find the next scheduled ATM channel
2. Distribute either:
   — An ICP cell
   — A filler cell (if ‘filler-only’ or ‘active’ with nothing scheduled in the APC)
   — A data cell (if a channel is scheduled in the APC, performing the appropriate segmentation task for the scheduled ATM channel, as determined by the channel’s AAL type)

The cells are distributed by writing the complete cells into circular transmit queues provided per link. These transmit queues function as ‘jitter buffers’, as they are used to decouple the transmit

rate of the TRL PHY from the transmit rate of the non-TRL PHYs in the group to allow for clocking differences between the PHYs.

At startup, the non-TRL links will transmit filler cells until their transmit queues have reached a minimum depth. In order to maintain less than the specified maximum +/-2.5 cell transmit timing differential for cells within an IMA frame, the TRL must exhibit the same behavior. Therefore, a four-cell transmit buffer is also maintained for the timing reference link. The timing reference link will only begin to pass ATM layer cells to its PHY after it has three cells in its buffer. Prior to this, it will send filler cells. This behavior will only be experienced at group start-up.

The TRL task will also implement the standard amount of stuffing on the TRL link by maintaining a counter. When this task has scheduled (2048/ M) ICP cells for the TRL, a TRL stuff event will be flagged and an indication of an upcoming stuff event will be signaled in the ICP LSI field. If a TRL stuff event is flagged when the TRL task triggers, then a stuff cell will be sent to the TRL's transmit queue, but no cells will be sent to the transmit queues of the non-TRL PHYs. This forces a standard amount of stuffing on the TRL, thereby reducing the effective data bit rate of the TRL to less than the minimum data clock rate allowed by the clock rate tolerance of the physical-layer standard. Therefore by definition, this effective data bit rate is achievable by the non-TRL links; the non-TRL links can either stuff less if their data clock rate is slower than the TRL, or stuff more if their data clock rate is faster than the TRL.

39.3.2.2 Non-TRL Operation

A request from a non-TRL PHY does not trigger any scheduling task. The cells for non-TRL links will already be supplied (by the TRL task) in its associated transmit queue. The TRL will simply read a cell out of its transmit queue and update the queue pointers.

If the transmit queue becomes too shallow (because this link's request rate is faster than the TRL), the link will flag that a stuff event is imminent. The link will signal an upcoming stuff event in the LSI field of its next ICP cell and will then flag that a stuff event is due. Having flagged this stuff event, the link will continue sending cells from its queue as normal until it reaches its next ICP cell, upon which it indicates a stuff event in the ICP cell and transmits it, but does not update the transmit queue pointers. When the link next requests a cell, the previous ICP cell is repeated (since the queue pointers were not updated). This process causes the transmit queue to deepen to the intended level.

At group start-up, instead of accessing its transmit queue, the link will send filler cells. This is to allow the transmit queues to reach their target steady-state depth. After the group start-up flag is cleared, normal operation as described above will commence.

39.3.2.3 Transmit Queue Operation Examples (ITC mode)

The following diagrams demonstrate the different cases of queue operation, and consequently justify the queue depth of 5 cells.

- The extraction pointer points to the queue entry that is currently being supplied to the PHY. This cell must be entirely ready when the PHY requests it.
- The insertion pointer points to the queue entry which will be filled next by the TRL process.

In the figures, note that the pointers and filled queue locations are just shown with respect to the overall queue depth available with the extraction pointer always shown at the bottom of the queue. This is done only for the purpose of ease of illustration. In reality, the transmit queues are circular queues in which the insertion and extraction pointers are continually rotating through the queue.

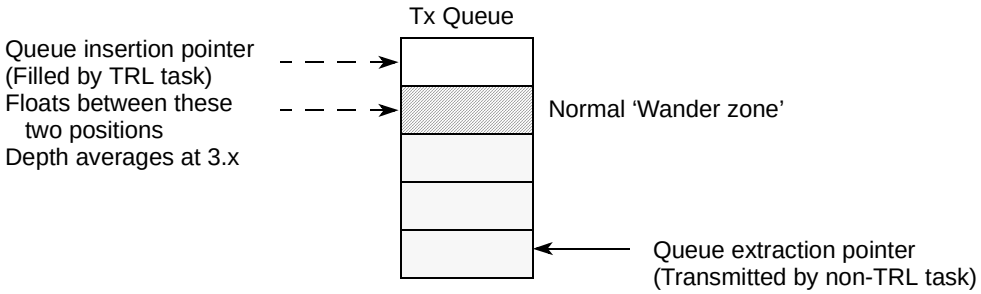


Figure 39-6. Transmit Queue Normal Operating State

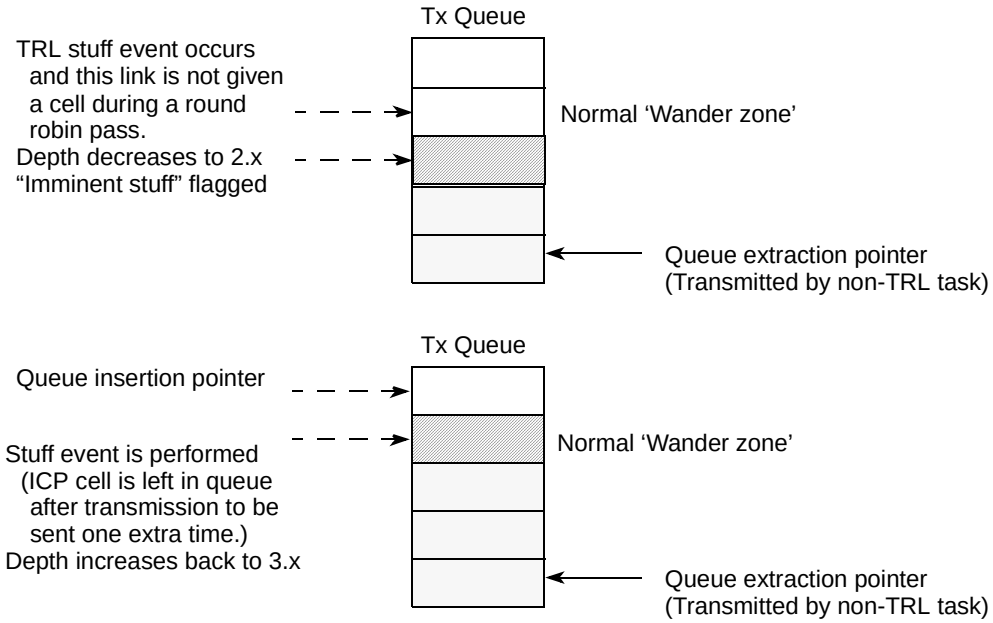


Figure 39-7. Transmit Queue Behavior: Link Clock Rate Same as TRL

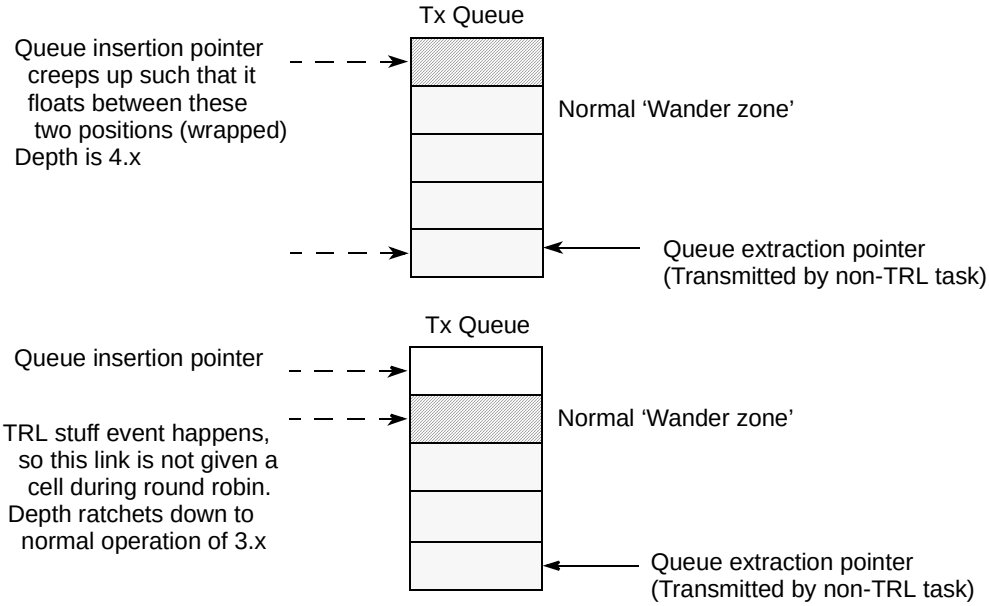


Figure 39-8. Transmit Queue Behavior: Link Clock Rate Slower than TRL

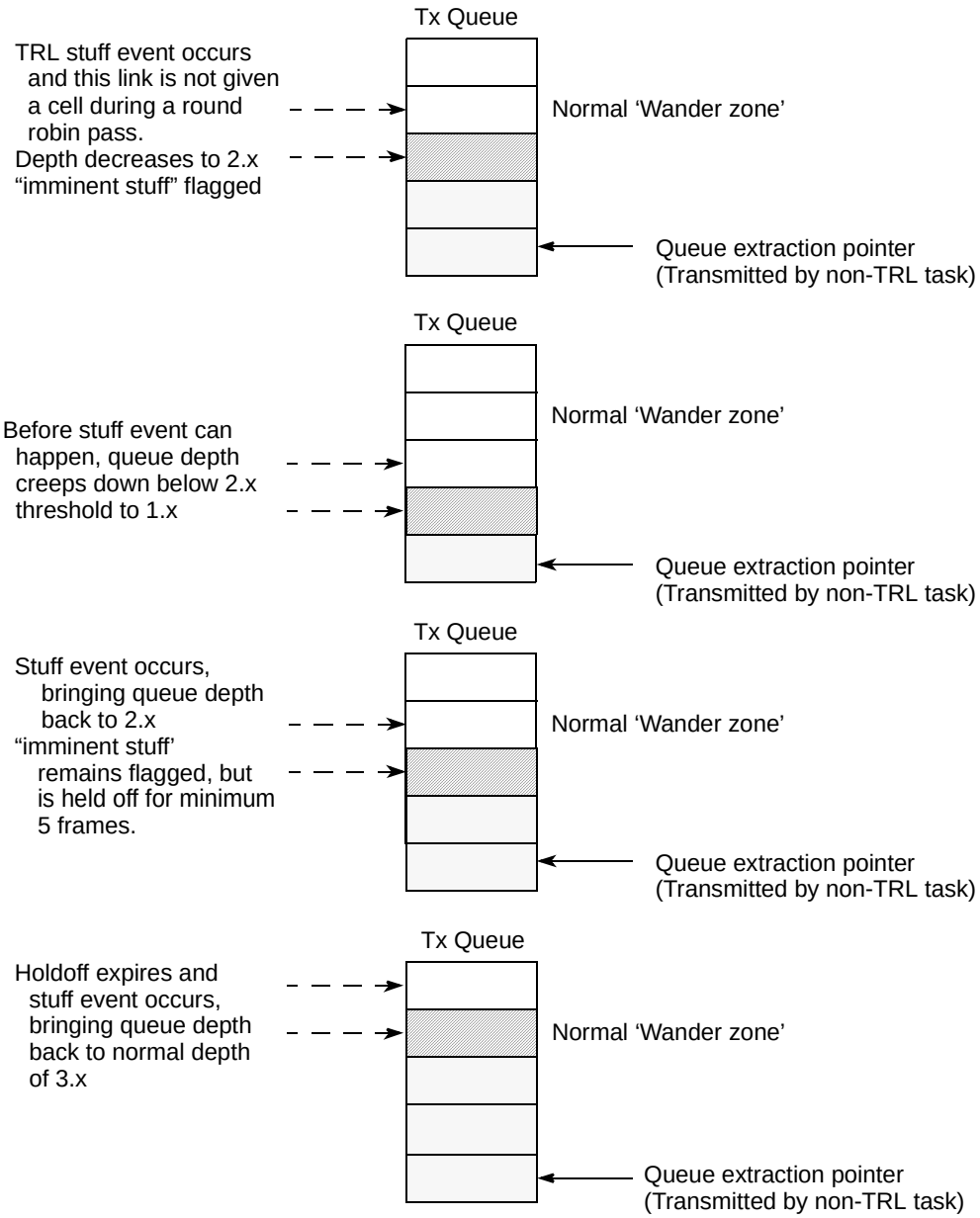


Figure 39-9. Transmit Queue Behavior: Link Clock Rate Faster than TRL, Worst-Case Event Sequence

39.3.2.4 Differences in CTC Operation

Overall, CTC operation is similar to ITC operation in task partitioning and overall structure. Differences are as follows:

1. Transmit buffers are still maintained for the non-TRL links, but these buffers will not ‘jitter’, since the transmit clocks are all the same. However, queue depths may vary slightly because PHY requests, while synchronous, will not necessarily come in simultaneously (that is, may have constant offsets) and will definitely not be serviced simultaneously.
2. The non-TRL tasks do not determine when to perform stuffing on the non-TRL links. When the TRL flags ‘imminent stuff’ on its own link, it will flag ‘imminent stuff’ on all of the non-TRL links as well. Thus, the non-TRL links will also stuff their links every 2048 cells.
3. The non-TRL queue depths are the same as the TRL’s queue (that is, four cells).

39.3.3 Receive Architecture

The receive task consists of three parts. The first part is for cell reception from the link. The second part provides the trigger for activating the cell processing task, including timing recovery if desired. The third part performs the actual cell processing (that is, passing cells to the ATM layer). The cell reception task services requests from the links (via the UTOPIA multi-PHY interface and the FCC), maintains the link state, and (if the link and group state dictates) writes the received cells into the link’s delay compensation buffer in external memory. The cell processing activation function coordinates passing cells from the cell reception task on an on-demand basis. The cell processing task extracts cells from the delay compensation buffers and passes them to the ATM layer for processing.

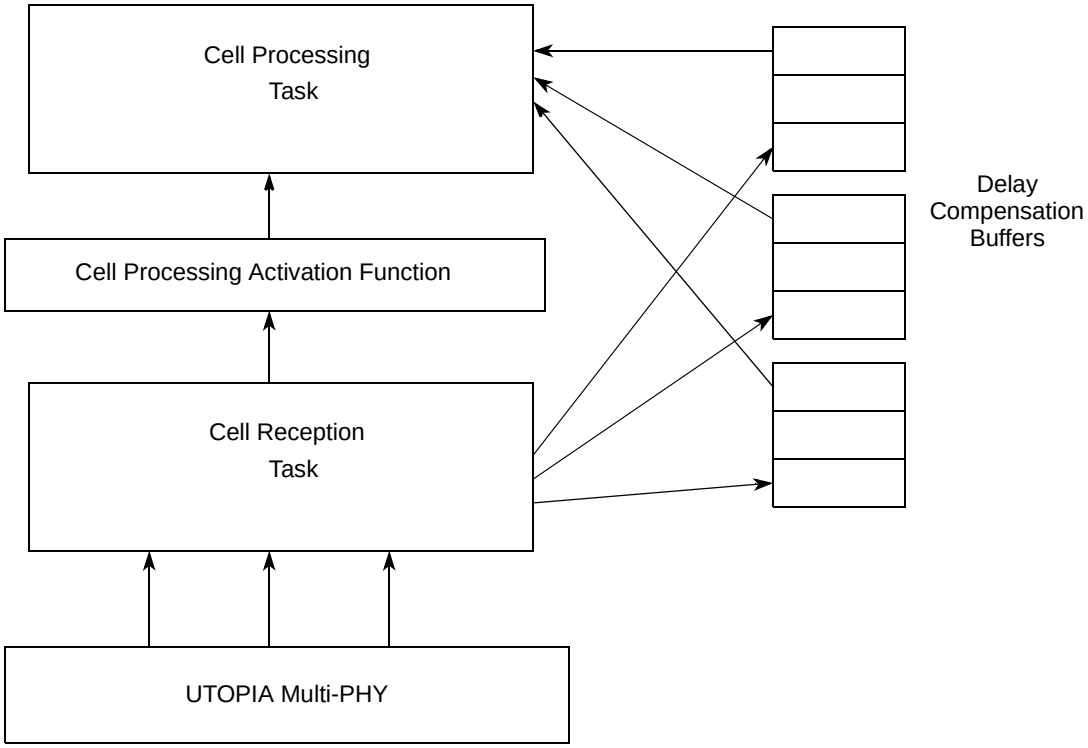


Figure 39-10. IMA Receive Task Interaction

39.3.3.1 Cell Reception Task

In the cell reception task, received ICP cells in which an SCCI change is noted, except for the first ICP received cell, are passed to a user-defined receive AAL0 channel to be processed by software. These received cells (and other event indications) are used by the software to initialize IMA links and groups, and to manage transitions between link and group states.

The cell reception task centers around a four-state link state machine. Tasks are performed within each state, but transitions between states are managed by software. The processing of received cells (both ICP cells and data cells) is determined by the state of the link.

The states are described as follows:

- Group Unassigned—This corresponds to a link which is known to be IMA, but for which no information is known (for example, IMA group number, IMA frame size). In this state, the receive task only screens the incoming cells for ICP cells and discards all others. ICP cells in which an SCCI change is noted are passed to a user-defined receive channel, where software must interpret their content in order to initialize the link and add it to the group. The link transitions to the ‘Link IFSM Unsynchronized state when the link’s ‘Group Assigned’ flag is set by software.
- Link IFSM Unsynchronized—In this state, the link state machine has not yet found or validated the frame boundary for this link. This state is initially entered after software

defines the link's M value and expected ICP cell format, then sets the 'Group Assigned' flag. This state can be subsequently re-entered as the result of a link defect. In this state, the link searches for an ICP cell, then looks for ICP cells in the expected position of subsequent frames in order to validate frame synchronization. Cells other than ICP cells are discarded. When the IFSM becomes synchronized, an interrupt to the host is generated. The link transitions to the 'Link Delay Unsynchronized' state when the link's group and link synchronization flags are set appropriately by software.

- Link Delay Unsynchronized—This state contains two separate operations, depending on the state of the group to which the link belongs. If the link enters this state as the result of group start-up, it performs the "new-group link delay synchronization algorithm". If the link enters this state while the group is already activated (per the link addition/slow recovery (LASR) procedure of the IMA standard), it performs the "added-link delay synchronization algorithm." As the result of either algorithm, an interrupt will be generated to the host when delay synchronization is achieved.
- No Link Defect—This is the state in which normal receive operation occurs after the link has been established and the link-state has been communicated appropriately to the far end. If the link is not inhibited at the group or link level, reception of ATM cells will occur. If the link or group is inhibited, non-ICP received cells will be replaced with filler cells. In the "no link defect" state, cells received (or their replacements) are written to the link's delay compensation buffer. The link will transition out of this state only as the result of an error or if reset by host software.

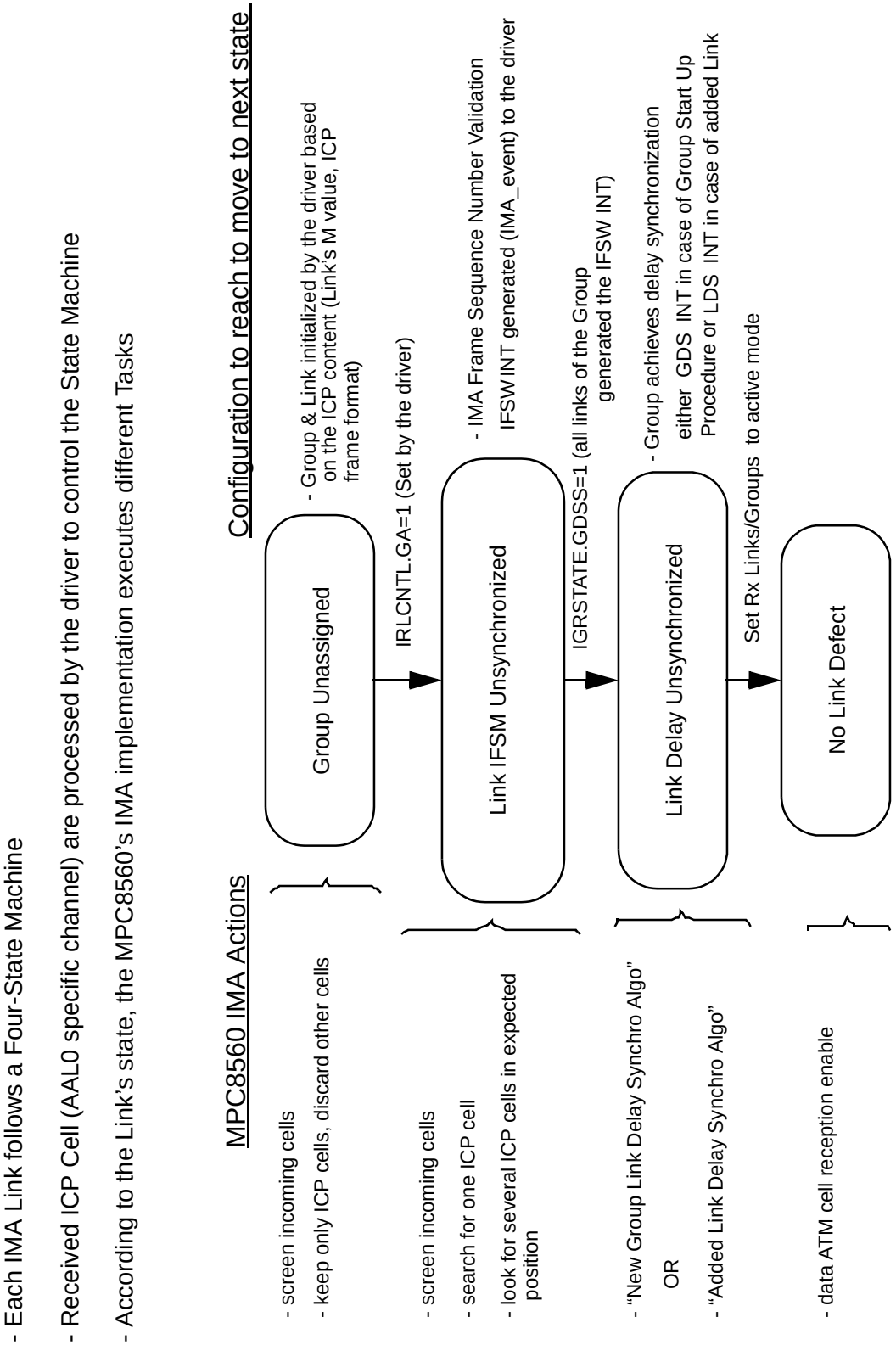


Figure 39-11. IMA Implementation: Receive Process

39.3.3.2 Cell Processing Activation Function

The cell processing activation function operates as on-demand cell processing.

39.3.3.2.1 On-Demand Cell Processing

In this mode, the cell processing activation function is null. The cell processing task is triggered directly by the cell reception task if a cell is written to a delay compensation buffer.

This mode is strictly demand-driven; there is no attempt to reconstruct an IMA Data Cell Rate (IDCR) from the IMA group at which to process incoming cells. Per the IMA specification, this is allowable under the following conditions:

- The IMA receiver is directly built into end equipment that directly terminates the ATM layer (that is, terminates all ATM connections), and
- The system is only capable of carrying services that either do not require CDV control (for example, some data services), or where the CDV is handled in some other way (for example, absorbed in a play-out buffer at the ATM layer connection termination).

The MPC8560 can qualify as such a system, if it terminates all ATM connections that it receives. The buffer-descriptors and external memory serve as a play-out buffer.

Furthermore, a system which does not terminate cells, but instead passes cells port-to-port, can also use this mode of operation if either of the following conditions are met:

- All cell streams are switched at the VC level only, and the VC's traffic type is one supported by the MPC8560's APC. In this case, the APC of the MPC8560 can be programmed to appropriately reshape the VC at the egress port, and therefore no cell delay variation (CDV) will be introduced.
- Some (or all) cell streams are switched at the VP level, but the switched VPs only carry traffic for which cell delay variation (CDV) within the bounds of an IMA round-robin distribution is tolerable. For example, if the IMA group consists of 8 DS1 links, then the maximum CDV introduced by this method would be 8 cell times, or approximately 2.2 ms

If the system meets the above qualifications, then this mode of operation is recommended, as it is the simplest and will yield overall better system performance (that is, this mode requires less CPM processing power).

39.3.3.3 Cell Processing Task

The cell processing task is triggered by the cell processing activation function. When the cell processing task is triggered, it will extract cells in-order from the delay compensation buffers. Cells extracted from the delay compensation buffers are processed per the standard MPC8560 ATM operation (that is, mapped into ATM channels and processed per the appropriate AAL or OAM function).

If the on-demand cell processing activation function is used, then when the cell processing task is triggered, it will extract cells in-order from the delay compensation buffers until either (1) no more are available, or (2) four cells have been extracted. The purpose of limiting the cell processing task to a maximum of four cells is to limit the maximum latency of servicing requests from the external PHYs. Note that, on the average, the receive process will deliver one cell to the ATM layer per cell reception.

39.4 IMA Programming Model

39.4.1 Data Structure Organization

Figure 39-12 shows the organization of IMA data structures.

The IMA data structures are organized as follows:

- Parameters at the FCC level determine common ATM parameters and location of the IMA Root Table.
- IMA Root Table includes parameters that are used by all IMA links of this FCC.
- IMA Group Receive Table and IMA Group Transmit Table entries include parameters that define the receive and transmit states and settings of the associated IMA group.
- IMA Link Receive Table and IMA Link Transmit Table entries include parameters that define the receive and transmit states and settings of the associated IMA link.

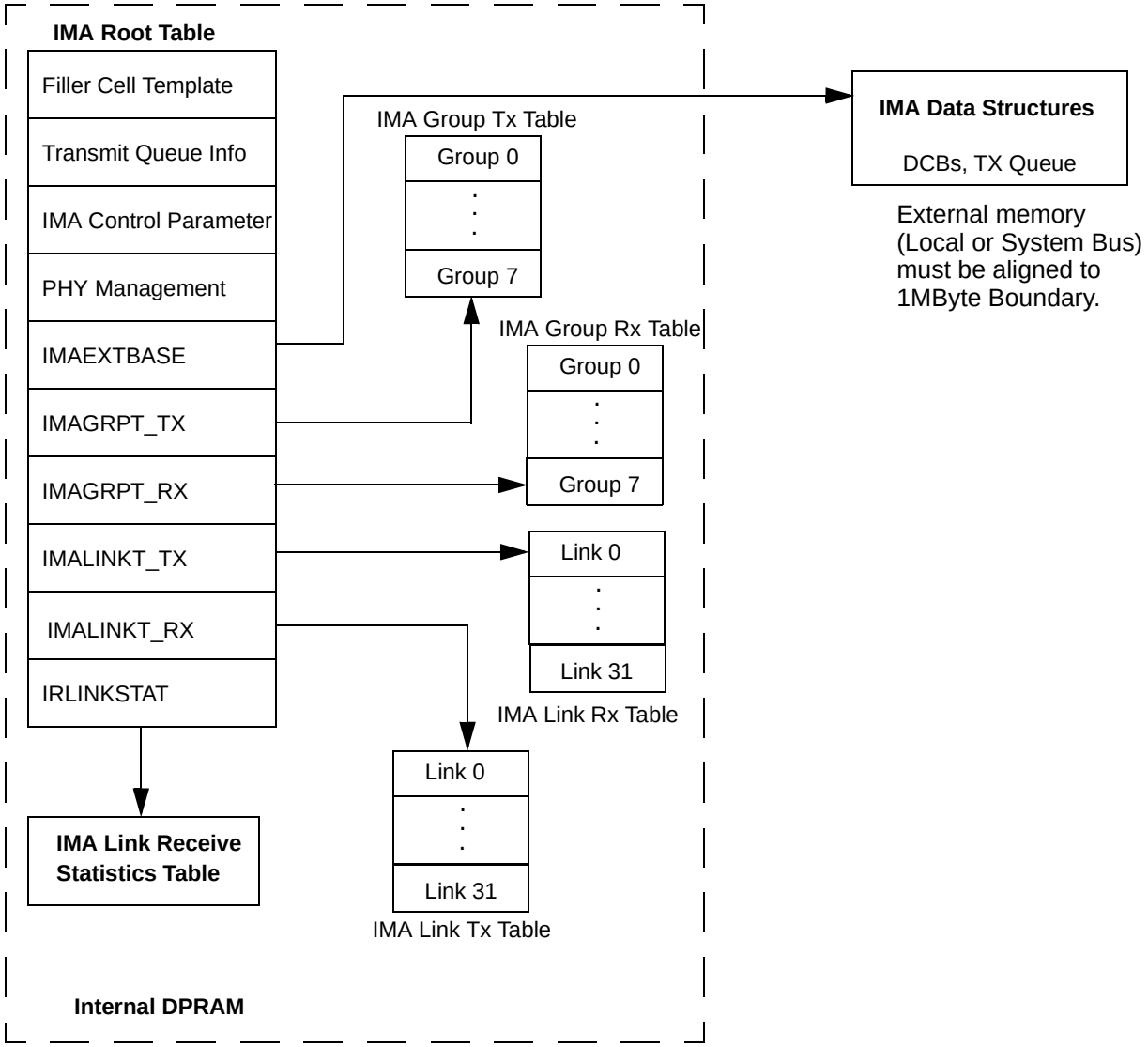


Figure 39-12. IMA Root Table Data Structures

39.4.2 IMA FCC Programming

39.4.2.1 FCC Registers

39.4.2.1.1 FPSMR*x*

The FCC protocol-specific mode register (FPSMR) for ATM operation is described in Section 35.13.2, “FCC Protocol-Specific Mode Register (FPSMR).” Refer to that section for information pertaining to IMA.

39.4.2.1.2 FTIRR*x*

For any PHY programmed in IMA mode (i.e. the corresponding bit in IMA PHY is set), the corresponding FTIRRx must be programmed to zero, for external rate mode. However, for PHYs 0-3, internal rate mode may be selected for non-IMA PHYs. Refer to Section 35.13.8, "FCC Transmit Internal Rate Register (FTIRRx)."

39.4.2.2 FCC Parameters

39.4.2.2.1 TCELL_TMP_BASE and RCELL_TMP_BASE

The TCELL_TMP_BASE and RCELL_TMP_BASE have the same definitions as for a standard FCC, in that they contain 64-byte aligned addresses of regions of DPRAM for temporary cell storage. However, the 4 bytes leading and 4 bytes following the region indicated by TCELL_TMP_BASE must also be reserved for use by the MPC8560's IMA implementation (thereby increasing its size to 60 bytes); and the 12 bytes following the region indicated by RCELL_TMP_BASE must also be reserve for use by the MPC8560's IMA implementation (thereby increasing its size to 64 bytes).

RCELL_TMP_BASE may be programmed to any 64-byte aligned address. TCELL_TMP_BASE must be programmed to a 64-byte aligned address terminating with 0x40 (that is, 0xnn40).

39.4.2.2.2 GMODE

IMA functionality in ROM is enabled using the GMODE register. Refer to Section 35.10.1.3, "Global Mode Entry (GMODE)."

39.4.2.3 IMA-Specific FCC Parameters

The following parameter must be programmed in the FCC parameter RAM page in addition to the standard FCC parameters for ATM.

Table 39-2. FCC Parameter RAM Additions

Offset	Name	Width	Description
0xEE	IMAROOT	Hword	Offset of IMA root table in DPRAM. Must be 128-byte aligned.

NOTE

IMAROOT must be programmed to a 128-byte aligned address terminating with 0x80 (that is, 0xnn80).

39.4.3 IMA Root Table

Table 39-3. IMA Root Table[1]

Offset	Name	Width	Description
0x04	**IMAFILLERHD**[2]	4 Bytes	Filler cell template. Used by the MPC8560 in transmission of filler cells. The cell is formatted as byte-swapped, and additionally the header is bitswapped. [This is due to hardware implementation, and does not imply the order of the transmission of the cell. The transmission of the cell is per the ATM standard.] Content should be: 0xD0000000 (header) 0x6A6A0001 or 0x6A6A0003 (for IMA Version 1.0 or 1.1) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0x6A6A6A6A (padding) 0xC6026A6A or 0xD9026A6A (for IMA Version 1.0 or 1.1)
0x00–0x2F	**IMAFILLERPLD**	48 Bytes	
0x30	FILLTAG	Byte	Tag indicating that the filler template is a filler cell. Program to zero.
0x31	**TQ_SIZE**	Byte	Transmit queue size. Recommended value is 0x18. Must be a multiple of 4. Refer to Section 39.4.6.1, “Transmit Queues” for more details
0x32	**TQ_TARGET**	Byte	Transmit queue target level. Recommended value is 0x0C. Must be a multiple of 4.
0x33	**TQ_THRESHOLD**	Byte	Transmit queue stuff threshold. Recommended value is 0x0C. Must be a multiple of 4.
0x34	RESERVED	Hword	Reserved
0x36	**IMACNTL**	Byte	IMA control parameter. Controls functions shared by all IMA groups for this FCC.
0x37	TMP_PCNT	Byte	MPC8560’s IMA managed parameter (pass count).
0x38	**RXPHYEN**	Word	Receive PHY enable. Bit array addressed by PHY address (for example, bit 0 corresponds to PHY 0). Setting a bit enables reception for the corresponding PHY. Must be used to enable/disable the corresponding PHY regardless of whether or not the PHY is defined as IMA in IMAPHY. All cells received by disabled PHYs are discarded. Note that the FCC must also be enabled in GFMR[ENR] for reception to occur. Bit 31 is reserved, and must be programmed to zero.
0x3C	**TXPHYEN**	Word	Transmit PHY enable. Bit array addressed by PHY address (for example, bit 0 corresponds to PHY 0). Setting a bit enables transmission for the corresponding PHY. Must be used to enable/disable the corresponding PHY regardless of whether or not the PHY is defined as IMA in IMAPHY. Only idle/unassigned cells are transmitted on disabled PHYs. Note that the FCC must also be enabled in GFMR[ENT] for transmission to occur. Bit 31 is reserved, and must be programmed to zero.

Table 39-3. IMA Root Table[1] (continued)

Offset	Name	Width	Description
0x40	**IMAPHY**	Word	Bit array addressed by PHY address (for example, bit 0 corresponds to PHY 0). Setting a bit defines the corresponding PHY to operate in IMA mode. Clearing a bit defines the corresponding PHY to operate as a normal (non-IMA) multi-PHY. Bit 31 is reserved, and must be programmed to zero.
0x44	**IMAEXTBASE**	Word	IMA external structure base pointer. Points to region in external memory where external IMA data structures are located. Must be aligned to a 1 MB boundary (that is, program bits 12–31 to zero).
0x48	**IMAGRPT_TX**	Hword	Offset of IMA group transmit table in DPRAM. Must be 16-byte aligned.
0x4A	**IMAGRPT_RX**	Hword	Offset of IMA group receive table in DPRAM. Must be 64-byte aligned.
0x4C	**IMALINKT_TX**	Hword	Offset of IMA link transmit table in DPRAM. Must be 32-byte aligned.
0x4E	**IMALINKT_RX**	Hword	Offset of IMA link receive table in DPRAM. Must be 32-byte aligned.
0x50	**IRLINKSTAT**	Hword	Offset of the optional IMA link receive statistics table in DPRAM. Must be 8-byte aligned.
0x52	TMP_LPTR_RX	Hword	MPC8560's IMA-managed parameter. Temporary storage of link table pointer.
0x54	TMP_LPTR_TX	Hword	MPC8560's IMA-managed parameter. Temporary transmit table pointer.
0x56	TMP_GPTR_TX	Hword	MPC8560's IMA-managed parameter. Temporary transmit group pointer.
0x58	TMP_GPORD_TX	Hword	MPC8560's IMA-managed parameter. Temporary transmit group order pointer.
0x5A	TMP_GPTR_RX	Hword	MPC8560's IMA-managed parameter. Temporary receive group pointer.
0X5C	TMP_RTRN_RX	Hword	MPC8560's IMA-managed parameter. Temporary return pointer.
0x5E	TMP_GPTR2_RX	Hword	MPC8560's IMA-managed parameter. Temporary receive group pointer 2.
0x60–0x7F	—	—	Reserved. Must be programmed to zero during initialization.

1 **Boldfaced** entries in the above table indicate parameters which must be initialized by the user. All other parameters are managed by the MPC8560 and should be initialized to zero unless otherwise stated.

2 IMAFILLERHD is located at the word which immediately precedes the 128-byte aligned region defined by IMAROOT. Thus it is located at offset - 0x04 from base of IMAROOT table.

39.4.3.1 IMA Control (IMACNTL)

The fields of the IMACNTL are shown in Figure 39-13.

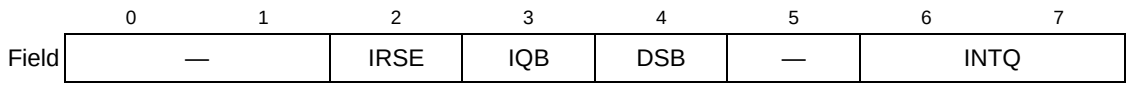


Figure 39-13. IMA Control (IMACNTL)

Table 39-4 describes the IMACNTL bit fields.

Table 39-4. IMACNTL Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	IRSE	IMA link receive statistics enable. If link receive statistics gathering is disabled, there is no need to initialize IRLINKSTAT or reserve space for the receive statistics table. 0 Link receive statistics gathering is disabled. 1 Link receive statistics gathering is enabled.
3	IQB	Interrupt queue bus. Defines on which bus the IMA interrupt queue is located. 0 On the system bus. 1 On the local bus.
4	DSB	Data structure bus. Defines on which bus the IMA external structure memory area is located. 0 On the system bus. 1 On the local bus.
5	—	Reserved
6–7	INTQ	Number of the ATM interrupt queue dedicated to IMA events. Note that these do not include ICP cell reception events; the handling of ICP cell reception events is programmed in the RCT of the ICP channel defined by RICPCH.

39.4.4 IMA Group Tables

The IMA group tables consist of multiple IMA group structures indexed by group number, which ranges from 0 to 7. The transmit and receive parameters are located in separate tables. The IMA group transmit table entries are 16 bytes long. The IMA group receive table entries are 64 bytes long. However, there is no need to reserve memory space in DPRAM for 8 receive IMA groups if less than 8 IMA groups are required. To conserve memory space used by these tables, it is best to add groups starting from group zero. Note also that group number is independent of the assignment of IMA ID.

39.4.4.1 IMA Group Transmit Table Entry

Table 39-5. IMA Group Transmit Table Entry [1]

Offset	Name	Width	Description
0x00	**IGTCNTL**	Byte	IMA group transmit control parameters.
0x01	IGTSTATE	Byte	IMA group transmit state. MPC8560's IMA-managed parameter. Must be initialized to zero at group start-up.
0x02	**TGRPORDER**	Hword	Offset of transmit group order table in DPRAM. Can be changed on the fly.

Table 39-5. IMA Group Transmit Table Entry (continued)[1]

Offset	Name	Width	Description
0x04	**TVPHYNUM**	Byte	Transmit Virtual PHY number. Maps this IMA transmit group to a virtual PHY number for the purpose of selecting an ATM pace controller (APC) table number for this IMA group. Note that this parameter must be unique; it must not conflict with the PHY number of any non-IMA PHYs or with the TVPHYNUM of any other IMA group. If there are any PHY numbers which will never be used in a system (that is, the actual PHY with the address does not exist), then TVPHYNUM should be selected from one of these PHYs. If no such PHY numbers are available (that is, the multi-PHY interface consists of the full 31 PHYs), then select TVPHYNUM from one of the PHY numbers of the PHYs within this IMA group.
0x05	TIFSN	Byte	Transmit IMA Frame Sequence Number (IFSN). MPC8560's IMA-managed parameter. Increments each time an IMA frame is transmitted, cycling from 0 through 255. Should be initialized to zero at group start-up.
0x06	TMCTR	Byte	Transmit IMA M counter. MPC8560's IMA-managed parameter. Tracks IMA frame boundaries. Increments once per round-robin distribution of cells to the transmit queues, cycling from 0 through M. Initialize to zero at group startup.
0x07	TRLSTFCNT	Byte	TRL stuff frame counter. MPC8560's IMA-managed parameter. Controls required stuffing on the TRL. Decrements each time an ICP cell is sent on the TRL, cycling from TRLSTFN through 0. A TRL stuff event occurs when it reaches zero. Initialize to the value of TRLSTFN at group start-up.
0x08	**TICPPTR**	Hword	Offset of transmit ICP cell payload template area in DPRAM. Must be 64-byte aligned. This parameter and the associated template area may only be changed when IGCNTL[ICPC]=IGTSTATE[ICPCA]. After changing TICPPTR, IGCNTL[ICPC] must be toggled.
0x0A	**TM**	Byte	Transmit IMA frame size. Program to 31, 63, 127, or 255 for frame sizes (M) of 32, 64, 128, or 256, respectively.
0x0B	**TRLSTFN**	Byte	TRL stuff frame number. Defines the number of IMA frames sent between TRL stuff events. For ITC operation IGTCNT[CTC] = 0, program TRLSTFN = 2048/M. For CTC operation IGTCNT[CTC] = 1, program TRLSTFN = (2048/M) – 1. Refer to Section 39.4.4.1.1, "IMA Group Transmit Control (IGTCNTL)."
0x0C	**TNUMLINKS**	Byte	Number of transmit links in the IMA group that are in the active state (ILTCNTL[TXSC] = 01). Used by the APC to scale the rescheduling parameters appropriately when rescheduling channels.
0x0D	RTSTPCNT	Byte	Real time-stamp subcounter. MPC8560's IMA-managed parameter, used by the APC. Initialize to zero at group start-up.

[1] **Boldfaced** entries must be initialized by the user. All other parameters initialize to zero.

39.4.4.1.1 IMA Group Transmit Control (IGTCNTL)

The fields of the IGTCNTL register are shown in Figure 39-14.

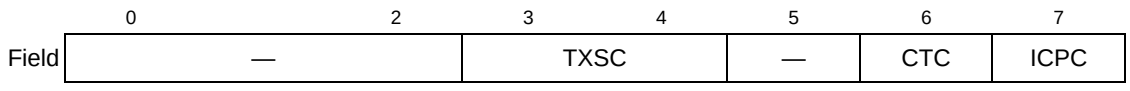


Figure 39-14. IMA Group Transmit Control (IGTCNTL)

Table 39-6 describes the IGTCNTL bit fields.

Table 39-6. IGTCNTL Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3–4	TXSC	Transmit status/control. Sets the transmit mode of the IMA group. 00 Filler mode. The IMA group transmits only ICP and filler cells, no data cells. 01 Active mode. The IMA group is capable of sending data cells. 1X Reserved.
5	—	Reserved
6	CTC[1]	Transmit clock mode for this IMA group. 0 Independent transmit clock (ITC) mode. 1 Common transmit clock (CTC) mode.
7	ICPC	ICP change flag. Maintains at least the minimum 2-frame spacing of ICP cell control/status changes. Initialize to zero at group start-up. Must be toggled by software whenever TICPPTR is changed. After two subsequent IMA frames are transmitted, the IGTSTATE[ICPCA] field will be changed to match IGTCNTL[ICPC]. When IGTCNTL[ICPC] equals IGTSTATE[ICPCA], changes to TICPPTR are allowed.

[1] Ensure transmit clock mode is not changed during IMA group startup to avoid erratic behavior.

39.4.4.1.2 IMA Group Transmit State (IGTSTATE)

The fields of the IGTSTATE register are shown in Figure 39-15.

	0	1	2	3	4	5	6	7
Field	TSTF	TIMSTF	—		ELX	—	ICH	ICPCA

Figure 39-15. IMA Group Transmit State (IGTSTATE)

Table 39-7 describes the IGTSTATE bit fields.

Table 39-7. IGTSTATE Field Descriptions

Bits	Name	Description
0	TSTF	TRL stuff flag. MPC8560's IMA-managed parameter. Indicates that the next ICP cell on the TRL will be part of a stuff event. Initialize to zero at group startup.
1	TIMSTF	TRL imminent stuff flag. MPC8560's IMA-managed parameter. Indicates that an upcoming stuff event will be signalled in the next ICP of the TRL (via LSI = 001). Initialize to zero at group startup.
2–3	—	Reserved
4	ELX	Early exit flag. MPC8560's IMA-managed parameter. Initialize to zero at group startup.
5	—	Reserved

Table 39-7. IGTSTATE Field Descriptions (continued)

Bits	Name	Description
6	ICH	ICP change holdoff. MPC8560's IMA-managed parameter. Initialize to zero at group startup.
7	ICPCA	ICP change allowed flag. MPC8560's IMA-managed parameter. See description of IGTCNTL[ICPC].

39.4.4.1.3 Transmit Group Order Table

The transmit group order table defines the order of the links in the round-robin distribution of cells to the links of the IMA group. The table consists of an array of bytes ordered from first link to last link, with each byte providing the PHY address of its associated link. The end of the table is indicated by an entry programmed to 0x1F.

This table alone defines the order of cell distribution. It is the responsibility of software to program the LID of the links in the IMA Link Table entries, and to program the group order table such that its order corresponds to the order of increasing LIDs within the group.

The format of a transmit group order table entry is shown in Figure 39-16.

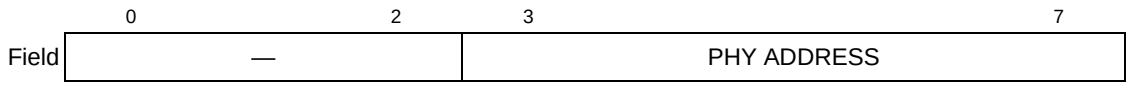


Figure 39-16. Transmit Group Order Table Entry

Table 39-8 describes the format of a transmit group order table entry.

Table 39-8. Transmit Group Order Table Entry Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3–7	PHY ADDRESS	PHY address (Up to 31 PHYs[0–30]) of the link transmitting in this position in the round-robin. A value of 0x1F in this field indicates the end of the group order table.

39.4.4.1.4 ICP Cell Templates

The ICP cell templates are areas of memory provided by software to the MPC8560's IMA implementation for construction of ICP cells for transmission. Software prepares the fields which are common to the group (that is, class B, C, D, and E parameters). The other (class A) parameters will be written to the appropriate fields by the MPC8560. The template is 53 bytes long and must be aligned on a 64-byte boundary.

Table 39-9 describes the format of a group order table entry. The ICP cell template is formatted as byte-swapped, and additionally the ICP cell header is bit swapped. (This is due to hardware implementation, and does not imply the order of transmission of the cell. The transmission of the

cell is per the ATM standard.) Reflecting this byte-swap, the offset column gives the offset in DPRAM from the ICP template base.

Table 39-9. ICP Cell Template [1]

Offset	Name	Width	Description
0x00	**ICP CELL HEADER**	Word	ICP cell header. Program to 0xD0000000.
0x04	ICP Cell Offset	Byte	MPC8560's IMA-managed area. This field is programmed dynamically.
0x05	IMA Frame Sequence Number	Byte	MPC8560's IMA-managed area. This field is programmed dynamically.
0x06	Cell ID and Link ID	Byte	MPC8560's IMA-managed area. This field is programmed dynamically.
0x07	**OAM LABEL**	Byte	IMA Version value. 1 IMA Version 1.0 3 IMA Version 1.1
0x08	**GROUP STATUS AND CONTROL**	Byte	Bits 7–4: Group State 0000 = Start-up, 0001 = Start-up-Ack, 0010 = Config-Aborted - Unsupported M, 0011 = Config-Aborted - Incompatible Group Symmetry, 0100 = Config-Aborted - Unsupported IMA Version, 0101, 0110 = Reserved for other Config-Aborted reasons in a future version of the IMA specification, 0111 = Config-Aborted - Other reasons, 1000 = Insufficient-Links, 1001 = Blocked, 1010 = Operational, Others: Reserved for later use in a future version of the IMA specification. Bits 3-2: Group Symmetry Mode 00 = Symmetrical configuration and operation, 01 = Symmetrical configuration and asymmetrical operation (optional), 10 = Asymmetrical configuration and asymmetrical operation (optional), 11 = Reserved Bits 1-0: IMA Frame Length (00: M=32, 01: M=64, 10: M=128, 11: M=256)
0x09	**IMA ID**	Byte	Bits 7–0: IMA ID
0x0A	**STATUS AND CONTROL CHANGE INDICATION (SCCI)**	Byte	Software must increment this field in the new ICP template whenever a new ICP template is created.
0x0B	Link Stuff Indication	Byte	MPC8560's IMA-managed area. This field is programmed dynamically.
0x0C	**RX TEST PATTERN**	Byte	Bits 7–0: Rx Test Pattern (value from 0 to 255)
0x0D	**TX TEST PATTERN**	Byte	Bits 7–0: Tx Test Pattern (value from 0 to 255)
0x0E	**TX TEST CONTROL**	Byte	Bits 7–6: Unused and set to 0 Bit 5: Test Link Command (0: inactive, l: active) Bits 4-0: Tx LID of test link (0 to 31)
0x0F	**TRANSMIT TIMING INFORMATION**	Byte	Bits 7–6: Unused and set to 0 Bit 5: Transmit Clock Mode: (0: ITC mode, 1: CTC mode) Bits 4–0: Tx LID of the timing reference (0 to 31)

Table 39-9. ICP Cell Template (continued)[1]

Offset	Name	Width	Description
0x10	**LINK 3 INFO**	Byte	Status and control of link with LID = 3
0x11	**LINK 2 INFO**	Byte	Status and control of link with LID = 2
0x12	**LINK 1 INFO**	Byte	Status and control of link with LID = 1
0x13	**LINK 0 INFO**	Byte	Status and control of link with LID = 0
0x14	**LINK 7 INFO**	Byte	Status and control of link with LID = 7
0x15	**LINK 6 INFO**	Byte	Status and control of link with LID = 6
0x16	**LINK 5 INFO**	Byte	Status and control of link with LID = 5
0x17	**LINK 4 INFO**	Byte	Status and control of link with LID = 4
0x18	**LINK 11 INFO**	Byte	Status and control of link with LID = 11
0x19	**LINK 10 INFO**	Byte	Status and control of link with LID = 10
0x1A	**LINK 9 INFO**	Byte	Status and control of link with LID = 9
0x1B	**LINK 8 INFO**	Byte	Status and control of link with LID = 8
0x1C	**LINK 15 INFO**	Byte	Status and control of link with LID = 15
0x1D	**LINK 14 INFO**	Byte	Status and control of link with LID = 14
0x1E	**LINK 13 INFO**	Byte	Status and control of link with LID = 13
0x1F	**LINK 12 INFO**	Byte	Status and control of link with LID = 12
0x20	**LINK 19 INFO**	Byte	Status and control of link with LID = 19
0x21	**LINK 18 INFO**	Byte	Status and control of link with LID = 18
0x22	**LINK 17 INFO**	Byte	Status and control of link with LID = 17
0x23	**LINK 16 INFO**	Byte	Status and control of link with LID = 16
0x24	**LINK 23 INFO**	Byte	Status and control of link with LID = 23
0x25	**LINK 22 INFO**	Byte	Status and control of link with LID = 22
0x26	**LINK 21 INFO**	Byte	Status and control of link with LID = 21
0x27	**LINK 20 INFO**	Byte	Status and control of link with LID = 20
0x28	**LINK 27 INFO**	Byte	Status and control of link with LID = 27
0x29	**LINK 26 INFO**	Byte	Status and control of link with LID = 26
0x2A	**LINK 25 INFO**	Byte	Status and control of link with LID = 25
0x2B	**LINK 24 INFO**	Byte	Status and control of link with LID = 24
0x2C	**LINK 31 INFO**	Byte	Program to 0x00.
0x2D	**LINK 30 INFO**	Byte	Status and control of link with LID = 30
0x2E	**LINK 29 INFO**	Byte	Status and control of link with LID = 29
0x2F	**LINK 28 INFO**	Byte	Status and control of link with LID = 28

Table 39-9. ICP Cell Template (continued)[1]

Offset	Name	Width	Description
0x30	CRC10	Hword	MPC8560's IMA-managed area. This field is programmed dynamically.
0x32	**END-TO-END CHANNEL**	Byte	Program to any value desired for end-to-end application-dependent signalling, or program to 0x00 if unused.
0x33	**UNUSED**	Byte	Program to 0x6A.
0x34	**TAG**	Byte	Internally-used tag value indicating that this is an ICP cell. Program to 0x80.

1 **Boldfaced** entries in the above table indicate parameters which must be initialized by the user. All other parameters are managed by the MPC8560 and should be initialized to zero unless otherwise stated.

39.4.4.2 IMA Group Receive Table Entry

Table 39-10. IMA Group Receive Table Entry[1]

Offset	Name	Width	Description
0x00	**IGRCNTL**	Byte	IMA group receive control parameters.
0x01	IGRSTATE	Byte	IMA group receive state. MPC8560's IMA-managed parameter. Initialize to zero at group startup.
0x02	**RIMAID**	Byte	Receive IMA ID. Program to the validated receive IMA ID value.
0x03	**IMAVER**	Byte	IMA version. Program to the validated IMA version value. 1 IMA Version 1.0 3 IMA Version 1.1
0x04	**RM**	Byte	Receive IMA frame size. Program to 31, 63, 127, or 255 for frame sizes of 32, 64, 128, or 256, respectively.
0x05	**RNUMLINKS**	Byte	Number of receive links in the IMA group that are in the active state (ILRCNTL[RXSC] = 01).
0x06	DCB_RM	Byte	Current cell (x out of RM cells in a frame) being processed by the reconstruction routine. MPC8560's IMA-managed parameter. Initialize to zero at group startup.
0x07	RSCCI	Byte	Receive IMA SCCI field. MPC8560's IMA-managed parameter. Holds the SCCI value of the last ICP cell received by this group.
0x08	DCBLNK	Hword	Pointer to link entry in group order table currently in use. MPC8560's IMA-managed parameter. Initialize to value of RGRPORDER0 at group startup.
0x0A	DCBX	Hword	Delay-compensation buffer extraction pointer. MPC8560's IMA-managed parameter. Initialize to zero at group startup.
0x0C	STALL_COUNT	Byte	Number of on-demand requests made for which there were no cells available in a link's DCB. MPC8560's IMA-managed parameter. Initialize to zero at group startup.
0x0D	REF_IFSN	Byte	Identifies the IFSN of the frame being processed by the 'reconstruction' routine. MPC8560's IMA-managed parameter.

Table 39-10. IMA Group Receive Table Entry[1] (continued)

Offset	Name	Width	Description
0x0E	GFP	Hword	Pointer to the start of the current frame being processed by the 'reconstruction' routine. MPC8560's IMA-managed parameter. Initialize to zero at group startup.
0x10	**RGRPORDER0**	Hword	Offset of receive group order table 0 in DPRAM.
0x12	**RGRPORDER1**	Hword	Offset of receive group order table 1 in DPRAM.
0x14	**ALPHABETA**	Byte	Alpha and beta parameters for IMA frame synchronization mechanism (IFSM). Bits 0–3: Alpha. Allowable range is 1–2; typical value is 2. Bits 4–7: Beta. Allowable range is 1–5; typical value is 2.
0x15	**GAMMA**	Byte	Gamma parameter for IMA frame synchronization mechanism (IFSM). Allowable range is 1–5; typical value is 1.
0x16	—	Hword	Reserved. Must be initialized to zero at group startup.
0x18	**REF_LINK**	Word	Bit array identifying which of the links (that is, PHY) in this group are enabled. Bit 0 corresponds to PHY 0, bit 30 corresponds to PHY 30. Software must set the corresponding bits to 1 so that the delay compensation process for the link(s) is started.
0x1C	LINK_ICP	Word	Bit array identifying which of the links in this group has received an ICP cell. A '1' in the corresponding link's bit position indicates that an ICP cell has been received. MPC8560's IMA-managed parameter. Initialize to zero at group startup.
0x20	LINK_DCB	Word	Bit array identifying which of the links in this group has started storing cells to its corresponding DCB. A '1' in the corresponding link's bit position indicates that cells have been stored in the DCB. MPC8560's IMA-managed parameter. Initialize to zero at group startup.
0x24	LINK_LD	Word	Bit array identifying which of the 'added' links in this group has a longer propagation delay (LD) than any of the existing links. A '1' in the corresponding link's bit position indicates that it has a longer propagation delay. MPC8560's IMA-managed parameter. Initialize to zero at group startup.
0x28	—	Byte	Reserved. Must be initialized to zero at group startup.
0x29	**RVPHYNUM**	Byte	Receive Virtual PHY number. Maps this IMA receive group to a virtual PHY number for the purpose of address mapping of received cells. Note that this parameter must be unique; it must not conflict with the PHY number of any non-IMA PHYs or with the RVPHYNUM of any other IMA group. If there are any PHY numbers which will never be used in a system (that is, the actual PHY with the address does not exist), then RVPHYNUM should be selected from one of these PHYs. If no such PHY numbers are available (that is, the multi-PHY interface consists of the full 31 PHYs), then select RVPHYNUM from one of the PHY numbers of the PHYs within this IMA group.

Table 39-10. IMA Group Receive Table Entry[1] (continued)

Offset	Name	Width	Description
0x2A	**STALL_THR**	Byte	Stall threshold. Used to detect stalled links when performing round-robin cell extraction from the delay compensation buffers (Dcbz). This is the number of cells which may be received without advancing the cell extraction pointer. The value is application-dependent and must be tuned by the user to the 'expected worst case.' Its value depends on the depth of queues and FIFOs in the complete transmit/receive path, and the 'burstiness' of the behavior of the FIFOs. Assuming very bursty FIFOs, it is approximately: STALL_THR = 2 × RNUMLINKS × (3 + RX_FIFO) where: a) RNUMLINKS is the number of links in the receive group order structure (regardless of link status). b) RX_FIFO is the depth of the receive FIFOs of the TC layer. c) 3 is the rounded value of the allowed transmit skew between links of a group (2.5, per the IMA standard). An optimal value for STALL_THR will be great enough to produce no link stall events in normal operation, but low enough to detect a failed link as quickly as possible.
0x2B	**IRGFS**	Byte	IMA Receive Group Frame Size Bits 0–5: Reserved Bits 6–7–GSC_M: Value of received ICP cell Group Status Control field (Bits 1:0) which determine the IMA frame size. This field must be programmed before links in the group are assigned
0x2C	LINK_LS	Word	Link stalled interrupt indication. Bit array identifying which links have issued a link stall (LS) interrupt. This parameter ensures that only one LS interrupt is generated per event. MPC8560's IMA-managed parameter. Initialize to zero at group startup.
0x30	LINK_DCBO	Word	Link DCB overflow interrupt indication. Bit array identifying which links have issued a link stall (LS) interrupt. This parameter ensures that only one LS interrupt is generated per event. MPC8560's IMA-managed parameter. Initialize to zero at group startup.
0x34–0x3F	—	3 Words	Reserved. Must be initialized to zero at group startup.

1 **Boldfaced** entries indicate parameters that must be initialized by the user. All other parameters are managed by the MPC8560 and should be initialized to zero unless otherwise stated.

39.4.4.2.1 IMA Group Receive Control (IGRCNTL)

The fields of the IGRCNTL register are shown in Figure 39-17.

	0	1	2	3	4	5	6	7
Field	GOTP	—		RXSC		—	—	NICPC

Figure 39-17. IMA Group Receive Control (IGRCNTL)

Table 39-11 describes the IGRCNTL bit fields.

Table 39-11. IGRCNTL Field Descriptions

Bits	Name	Description
0	GOTP	Group order table pointer. Defines which group order table pointer (RGRPORDER0 or RGRPORDER1) will be used for the cell extraction round-robin. Initialize to zero at group startup.
1–2	—	Reserved, initialize to zero.
3–4	RXSC	Receive status/control. Sets the receive mode of the IMA group. 00 Filler mode. The IMA group processes only ICP cells. Data cells are replaced with filler cells. 01 Active mode. The IMA group is capable of receiving data cells. 1X Reserved. Defaults to Filler Mode.
5–7	—	Reserved, initialize to zero.

39.4.4.2.2 IMA Group Receive State (IGRSTATE)

The fields of the IGRSTATE register are shown in Figure 39-18.

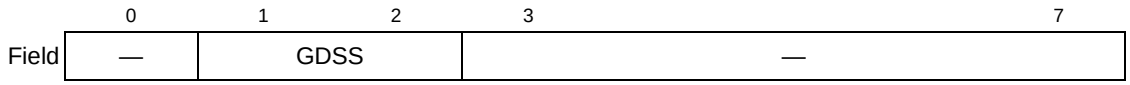


Figure 39-18. IMA Group Receive State (IGRSTATE)

Table 39-12 describes the IGRSTATE bit fields.

Table 39-12. IGRSTATE Field Descriptions

Bits	Name	Description
0	—	Reserved, initialize to zero at group startup.
1–2	GDSS	Group delay synchronization state. Initialize to zero at group startup. Must be changed by software to 01 after sufficient links have achieved frame synchronization. Subsequently managed by the MPC8560. 00 Group delay synchronization process inhibited. 01 Group delay synchronization process enabled. 10 Group delay synchronization process in progress. 11 Group delay synchronized.
3–7	—	Reserved, initialize to zero at group startup.

39.4.4.2.3 IMA Receive Group Frame Size

The fields of the IRGFS register are shown in Figure 39-19.

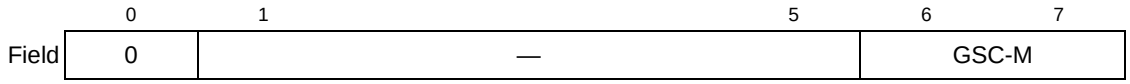


Figure 39-19. IMA Receive Group Frame Size (IGRSTATE)

Table 39-13 describes the IRGFS bit fields.

Table 39-13. IRGFS Field Descriptions

Bits	Name	Description
0	0	Reserved, initialize to zero.
1–5	—	Reserved, initialize to zero.
6–7	GSC-M	IMA Receive Group Frame Size Value of received ICP cell Group Status Control field (Bits 1–0) which determine the IMA frame size. This field must be programmed before links in the group are assigned

39.4.4.2.4 Receive Group Order Tables

The receive group order tables define the order of the links in the round-robin extraction of cells to the links of the IMA group. The table consists of an array of bytes ordered from first link to last link, with each byte providing the PHY address of its associated link. The end of the table is indicated by an entry programmed to 0x1F.

Two group order tables are used in order to allow for on-the-fly changes (that is, to add or remove links), while making certain that the changes only occur at the end of a round-robin cycle. IGRCNTL[GOTP] indicates which group order table pointer (and therefore, group table) is currently in use. Changes should be made to the group order table not in use, and then IGRCNTL[GOTP] should be toggled. When the current round-robin cell extraction process completes, the next process will use the new table.

This table alone defines the order of cell extraction from the delay compensation buffers. It is the responsibility of software to read the LIDs of the links in the group from the received ICP cells during group startup, and to program the group order table such that its order corresponds to the order of increasing LIDs within the group.

The format of a receive group order table entry is shown in Figure 39-20.

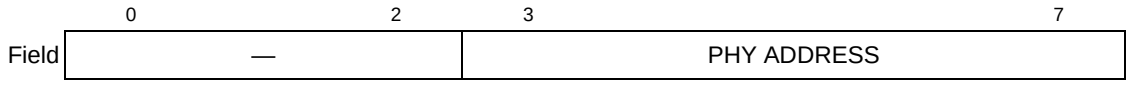


Figure 39-20. Receive Group Order Table Entry

Table 39-14 describes the format of a receive group order table entry.

Table 39-14. Receive Group Order Table Entry Field Descriptions

Bits	Name	Description
0–2	—	Reserved, initialize to zero.
3–7	PHY ADDRESS	PHY address (up to 31 PHYs[0–30]) of the link transmitting in this position in the round-robin. A value of 0x1F in this field indicates the end of the group order table.

39.4.5 IMA Link Tables

The IMA link tables consist of multiple IMA group structures indexed by the PHY address of their corresponding PHYs. The transmit and receive parameters are located in separate tables. The IMA group transmit and receive table entries are each 32 bytes long.

Note that to conserve memory space consumed by these tables, it is best to group the addresses of the PHYs that are assigned as IMA. (For example, if out of 8 total links only 4 are IMA, then optimally these would be links 0 through 3.)

39.4.5.1 IMA Link Transmit Table Entry

Table 39-15. IMA Link Transmit Table Entry[1]

Offset	Name	Width	Description
0x00	**ILTCNTL**	Byte	IMA link transmit control parameters.
0x01	ILTSTATE	Byte	IMA link transmit state. MPC8560's IMA-managed parameter. Initialize to zero at link startup.
0x02	**LICPOS**	Byte	Link ICP offset. Determines the position of the ICP cell within the IMA frame. Program in the range 0 to M-1. See the IMA specification for recommended methods for programming this field.
0x03	**ILID**	Byte	IMA link ID. Formatted per the Cell ID and Link ID field of an ICP cell. Bit 0: 1 (indicating ICP cell) Bits 1–2: Not Used, set to zero Bits 3–7: Program to software-assigned LID. (Note: This value is only used by the MPC8560 to format ICP cells for this link, it is not used to determine round-robin transmission order. That function is performed by the group order table.)
0x04	ITSEC	Word	IMA transmit stuff event counter. While ILTCNTL[SES] = 0, increments each time a stuff event is performed on this link. Initialize to zero at link startup.
0x08	**ITQSP**	Hword	IMA link transmit queue start pointer. This parameter forms bits 12–27 of the pointer; bits 0–11 are from IMAEXTBASE, and bits 28–31 are zero. Refer to Section 39.4.6.1, "Transmit Queues."
0x0A	**ITQEP**	Hword	IMA link transmit queue end pointer. This parameter forms bits 12–27 of the pointer; bits 0–11 are from IMAEXTBASE, and bits 28–31 are zero. Points to the last cell in the transmit queue. Program this parameter to ITQSP+TQ_SIZE-4. Refer to Section 39.4.6.1, "Transmit Queues."
0x0C	**ITQFP**	Hword	IMA link transmit queue fill pointer. This parameter forms bits 12–27 of the pointer; bits 0–11 are from IMAEXTBASE, and bits 28–31 are zero. Initialize this to the value of ITQSP. Refer to Section 39.4.6.1, "Transmit Queues."
0x0E	**ITQXP**	Hword	IMA link transmit queue extract pointer. This parameter forms bits 12–27 of the pointer; bits 0–11 are from IMAEXTBASE, and bits 28–31 are zero. Initialize this to the value of ITQSP. Refer to Section 39.4.6.1, "Transmit Queues."

Table 39-15. IMA Link Transmit Table Entry[1] (continued)

Offset	Name	Width	Description
0x10	**ITINTMSK**	Byte	IMA transmit interrupt mask. Has the same format as the upper byte of the IMA interrupt queue entry. Setting a bit enables the associated interrupt; clearing a bit masks it. For group-related events, only the mask register for the TRL is referenced.
0x11	ITINTSTAT	Byte	IMA transmit interrupt status. Indicates the status of transmit interrupt events.
0x12	LSHC	Byte	Stuff holdoff counter. Maintains the minimum five-frame spacing between stuff events for non-TRL links. Must be initialized to zero. Set to 4 by the MPC8560 after a stuff event, and decrements after each ICP cell is sent (saturating at zero). Signaling of ‘imminent stuff’ (and subsequent stuff events) will not occur while this counter is non-zero.

1 **Boldfaced** entries indicate parameters that must be initialized by the user. All other parameters are managed by the MPC8560 and should be initialized to zero unless otherwise stated.

39.4.5.1.1 IMA Link Transmit Control (ILTCNTL)

The fields of the ILTCNTL register are shown in Figure 39-21.

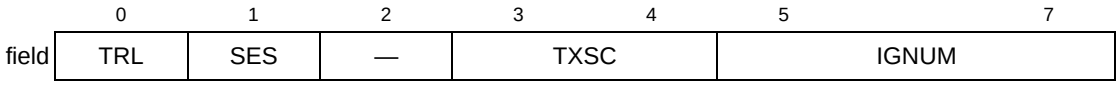


Figure 39-21. IMA Link Transmit Control (ILTCNTL)

Table 39-16 describes the ILTCNTL bit fields.

Table 39-16. ILTCNTL Field Descriptions

Bits	Name	Description
0	TRL	Defines this link as the timing reference link (TRL) of the group. 0 This link is not the TRL. 1 This link is the TRL. Note: One and only one link of the group must be programmed as the TRL. If zero or more than one links are defined as TRL, erratic operation will occur.
1	SES	Severely errored seconds. Set by software when a severely errored second indication is detected. When set, causes the transmit stuff event counter to stop counting.
2	—	Reserved, initialize to zero.
3–4	TXSC	Transmit status/control. Sets the transmit mode of the IMA link. 00 Filler mode. The IMA link transmits only ICP and filler cells, no data cells. 01 Active mode. The IMA link is capable of sending data cells. 1X Reserved.
5–7	IGNUM	Determines to which IMA group this link belongs. Contains the number of the link’s associated IMA Group Table entry. Note: This value need not be the same as the group’s IMA ID; the IMA ID is programmed separately in the group’s ICP cell template.

39.4.5.1.2 IMA Link Transmit State (ILTSTATE)

The fields of the ILTSTATE register are shown in Figure 39-22

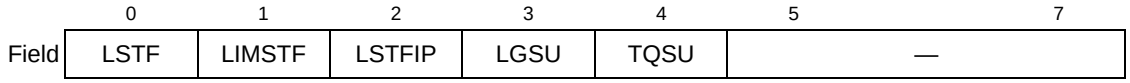


Figure 39-22. IMA Link Transmit State (ILTSTATE)

Table 39-17 describes the ILTSTATE bit fields.

Table 39-17. ILTSTATE Field Descriptions

Bits	Name	Description
0	LSTF	Link stuff flag. MPC8560's IMA-managed parameter. Indicates that the next ICP cell on this link will be part of a stuff event. Initialize to zero at link startup.
1	LIMSTF	Link imminent stuff flag. MPC8560's IMA-managed parameter. Indicates that an upcoming stuff event will be signalled in the next ICP of this link (via LSI = 001). Initialize to zero at link startup.
2	LSTFIP	Link stuff-in-progress flag. MPC8560's IMA-managed parameter. Indicates that the next cell on this link will be the second cell of a stuff event. Initialize to zero at link startup.
3	LGSU	Link/group startup flag. MPC8560's IMA-managed parameter. Will be set by the MPC8560 after the link's transmit queue has reached target average depth of 3 cells. While this bit is cleared, the link will transmit only filler cells. Initialize to zero at link startup.
4	TQSU	Transmit queue startup flag. MPC8560's IMA-managed parameter. Will be set by the MPC8560 after the first cell has been sent to the transmit queue. Initialize to zero at link startup.
5–7	—	Reserved, initialize to zero.

39.4.5.1.3 IMA Transmit Interrupt Status (ITINTSTAT)

The fields of the ITINTSTAT register are shown in Figure 39-23.

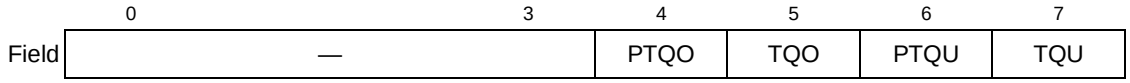


Figure 39-23. IMA Transmit Interrupt Status (ITINTSTAT)

Table 39-18 describes the ITINTSTAT bit fields.

Table 39-18. ITINTSTAT Field Descriptions

Bits	Name	Description
0–3	—	Reserved, initialize to zero.
4	PTQO	Persistent transmit queue overflow. Set when a transmit queue overflow occurs for two cells in a row. Further transmit queue overflow events are masked when this bit is set, in order to avoid a ‘flood’ of interrupts. This bit can be used to distinguish a temporary overflow condition (which could be caused by a rate differential between the TRL and non-TRL link which was out of compensatory range), or a persistent overflow condition (which could be caused by a more permanent condition, such as a failure of this link’s PHY device). Initialize to zero at link startup.
5	TQO	Transmit queue overflow. Set when a transmit queue overflow occurs; cleared when a cell is successfully transmitted. As this bit is set or cleared on a cell-by-cell basis, it may no longer be set when software reads it if the overflow condition was only temporary. PTQO should be used instead to distinguish between persistent or temporary underrun conditions.
6	PTQU	Persistent transmit queue underrun. Set when a transmit queue underrun occurs for two cells in a row. Further transmit queue underrun events are masked when this bit is set, in order to avoid a ‘flood’ of interrupts. This bit can be used to distinguish a temporary underrun condition (which could be caused by a rate differential between the TRL and non-TRL link which was out of compensatory range), or a persistent underrun condition (which could be caused by a more permanent condition, such as a TRL failure). Initialize to zero at link startup.
7	TQU	Transmit queue underrun. Set when a transmit queue underrun occurs; cleared when a cell is successfully transmitted. As this bit is set or cleared on a cell-by-cell basis, it may no longer be set when software reads it if the underrun condition was only temporary. PTQU should be used instead to distinguish between persistent or temporary underrun conditions. Initialize to zero at link startup.

39.4.5.2 IMA Link Receive Table Entry

Table 39-19. IMA Link Receive Table Entry[1]

Offset	Name	Width	Description
0x00	**ILRCNTL**	Hword	IMA link receive control parameters.
0x02	ILRSTATE	Hword	IMA link receive state. MPC8560’s IMA-managed parameter. Initialize to 0x0040 at link startup.
0x04	**ILID**	Byte	IMA link ID. Formatted per the Cell ID and Link ID field of an ICP cell. Bit 0: 1 (indicating ICP cell) Bits 1–2: Program to zero Bits 3–7: Program to validated LID for this link [Note: This value is only used by the MPC8560 to validate incoming ICP cells for this link, it is not used to determine round-robin transmission order. That function is performed by the group order table.]
0x05	RMCTR	Byte	Receive M counter. MPC8560’s IMA-managed parameter.
0x06	LRIFSN	Byte	Receive IFSN counter. MPC8560’s IMA-managed parameter.

Table 39-19. IMA Link Receive Table Entry[1] (continued)

Offset	Name	Width	Description
0x07	DFC	Byte	Number of frames to discard on a this link until it is caught up with the other links in this group (long propagation delay). MPC8560's IMA-managed parameter.
0x08	**DCBSP**	Hword	IMA link delay compensation buffer start pointer. This parameter forms bits 12–27 of the pointer; bits 0–11 are from IMAEXTBASE, and bits 28–31 are zero. Refer to Section 39.4.6.2, "Delay Compensation Buffers (DCB)," for more details.
0x0A	**DCBEP**	Hword	IMA link delay compensation buffer end pointer. This parameter forms bits 12–27 of the pointer; bits 0–11 are from IMAEXTBASE, and bits 28–31 are zero. Refer to Section 39.4.6.2, "Delay Compensation Buffers (DCB)," for more details.
0x0C	**DCBFP**	Hword	IMA link delay compensation buffer fill pointer. MPC8560's IMA-managed parameter. This parameter forms bits 12–27 of the pointer; bits 0–11 are from IMAEXTBASE, and bits 28–31 are zero. Initialize to DCBSP at link startup.
0x0E	**DCBRP**	Hword	IMA link delay compensation buffer read pointer. Only used during group delay synchronization and link addition. MPC8560's IMA-managed parameter. This parameter forms bits 12–27 of the pointer; bits 0–11 are from IMAEXTBASE, and bits 28–31 are zero.
0x10	**RICPCH**	Hword	Receive ICP channel number. ATM receive channel number to which received ICP cells are sent.
0x12	**IRINTMSK**	Byte	IMA receive interrupt mask. Has the same format as the IMA interrupt queue entry, however only receive-related bits are relevant. Setting a bit enables the associated interrupt; clearing a bit masks it. For group-related events, only the mask register for the TRL is referenced.
0x13	**LICPOS**	Byte	Link ICP offset. Program to the ICP offset validated for this link.
0x14	IRSEC	Word	IMA receive stuff event counter. Increments each time a stuff event is received on this ink. Initialize to zero at link startup.
0x18	ANOMALY_CTR	Byte	Anomaly counter. MPC8560's IMA-managed parameter. Initialize to zero at link startup.
0x19	ALPHABETA_CTR	Byte	Alpha/beta counter. MPC8560's IMA-managed parameter. Initialize to zero at link startup.

Table 39-19. IMA Link Receive Table Entry[1] (continued)

Offset	Name	Width	Description
0x1A	GAMMA_CTR	Byte	Gamma counter. MPC8560's IMA-managed parameter. Initialize to zero at link startup.
0x1C	DEFECT_CTR	Word	Defect counter. This counter is active while the link is in the Loss-of-IMA-Frame (LIF) state and is used to ensure IFSD interrupts are generated for every GAMMA+2 frames. Software can use the period interrupt issued by this counter in order to determine if the link is taking too long to synchronize. The DEFECT_CTR is active before IFSM reaches SYNC. It starts counting from the first cell received and will count from 0 to (GAMMA+2) × M. When it reaches (GAMMA+2) × M an IFSD interrupt is generated and the counter is reset. Upon reception of the next cell it starts to count again and subsequent interrupts are generated. MPC8560's IMA-managed parameter. Initialize to zero at link startup.

1 **Boldfaced** entries indicate parameters that must be initialized by the user. All other parameters are managed by the MPC8560 and should be initialized to zero unless otherwise stated.

39.4.5.2.1 IMA Link Receive Control (ILRCNTL)

The fields of the ILRCNTL register are shown in Figure 39-24

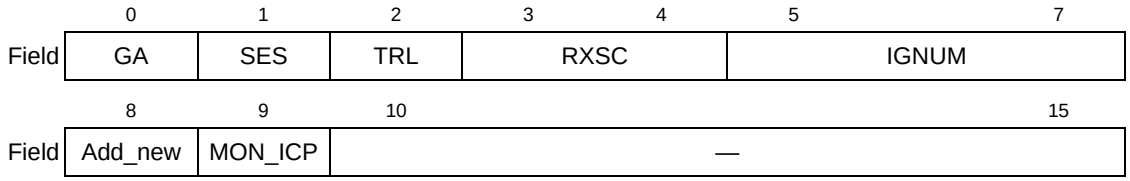


Figure 39-24. IMA Link Receive Control (ILRCNTL)

Table 39-20 describes the ILRCNTL bit fields.

Table 39-20. ILRCNTL Field Descriptions

Bits	Name	Description
0	GA	Group assigned flag. Set by software after group parameters have been validated. 0 Group unassigned. 1 Group assigned.
1	SES	Severely errored seconds. Set by software when a severely errored second indication is detected. When set, causes the receive stuff event counter and ICP violation counter to stop counting.
2	TRL	Identifies this link in the group as the Timing Reference Link. (Note: Only one link per group can be selected as the TRL.)

Table 39-20. ILRCNTL Field Descriptions (continued)

Bits	Name	Description
3–4	RXSC	Receive status/control. Sets the receive mode of the IMA link. 00 Filler mode. The IMA link processes only ICP cells. Data cells are replaced with filler cells. 01 Active mode. The IMA link is capable of receiving data cells. 10 Dropped. Must be set by the user during the process of dropping the link. Dropped links are treated filler mode until they are switched out of the receive round-robin (that is, data cells are replaced with filler cells). 11 Reserved.
5–7	IGNUM	Determines to which IMA group this link belongs. Contains the number of the link's associated IMA Group Table entry. Note: This value need not be the same as the group's IMA ID; the IMA ID is programmed separately in the receive group's RIMAID field.
8	ADD_NEW	Identifies this link as a new/added link to a group. Software must flip (0 to 1 or 1 to 0) this bit ONLY when adding a link to an existing group that is operational. If ADD_NEW = ILRSTATE[ADD_NEW_M] = 0, set ADD_NEW to 1 to indicate this is an added link. If ADD_NEW = ILRSTATE[ADD_NEW_M] = 1, set ADD_NEW to 0 to indicate this is an added link. Initialize to zero.
9	MON_ICP	Setting this bit will allow changed ICP cells received on this link to be passed on to the ATM layer (defined channel). The user must monitor at least one link in order for ICP cells to be passed on to the ATM layer.
10–15	—	Reserved, initialize to zero

39.4.5.2.2 IMA Link Receive State (ILRSTATE)

The fields of the ILRSTATE register are shown in Figure 39-25

	0	1 2	3	4	5	6	7
Field	ADD_NEW_M	IFSS	MASK_ERR	STFEX	SU	CHK_NXT	DATA_ERR

	8	9 10	11	12	13	14 15
Field	ICP_CELL	FSES	SYNC_DEFECT	STFIP	DL	—

Figure 39-25. IMA Link Receive State (ILRSTATE)

Table 39-21 describes the ILRSTATE bit fields.

Table 39-21. ILRSTATE Field Descriptions

Bits	Name	Description
0	ADD_NEW_M	'New Link' shadow bit. MPC8560 IMA-managed parameter. Initialize to zero at link startup.
1–2	IFSS[1]	IMA frame synchronization state. MPC8560 IMA-managed parameter. Initialize to zero at link startup. 00 IMA hunt. 01 IMA presync. 1x IMA sync.
3	MASK_ERR	Mask Error. MPC8560 IMA-managed parameter. Initialize to zero at link startup.

Table 39-21. ILRSTATE Field Descriptions (continued)

Bits	Name	Description
4	STFEX	Stuff cell expected. MPC8560 IMA-managed parameter. Initialize to zero at link startup.
5	SU	Start Up. MPC8560 IMA-managed parameter. Initialize to zero at link startup.
6	CHK_NXT	Check Next. MPC8560 IMA-managed parameter. Initialize to zero at link startup.
7	DATA_ERR	Data Error - Parity/CRC. MPC8560 IMA-managed parameter. Initialize to zero at link startup.
8	ICP_CELL	Current Cell is an ICP Cell. MPC8560 IMA-managed parameter. Initialize to zero at link startup.
9–10	FSES[1]	Frame Synchronization Error State. MPC8560 IMA-managed parameter. Initialize to 10 at link startup. 00 IMA working. 01 Out-of-IMA frame anomaly. 1x Loss-of-IMA frame defect.
11	SYNC_DEFECT	Synchronization Defect. MPC8560 IMA-managed parameter. Initialize to zero at link startup.
12	STFIP	Stuff In-Progress flag. MPC8560 IMA-managed parameter. Initialize to zero at link startup.
13	DL	Dropped link. MPC8560 IMA-managed parameter. Initialize to zero at link startup.
14–15	—	Reserved, initialize to zero

[1] Enable these parameters to their default values during IMA initialization to avoid spurious IMA exceptions in the IMA interrupt queues.

39.4.5.3 IMA Link Receive Statistics Table

The IMA link receive statistics table is optional. It is enabled globally for this FCC via IMACNTL[IRSE]. The base of this table is determined by IRLINKSTAT. Entries in the table are indexed by the link number of the associated link. The format for the IMA link receive statistic table entries is shown in Table 39-22

Table 39-22. IMA Link Receive Statistics Table Entry

Offset	Name	Width	Description
0x00	ICPVIOL	Word	IMA receive ICP violation event counter. While ILRCNTL[SES] = 0, increments each time an errored, invalid, or missing ICP cell is received on this link. Initialize to zero at link startup.
0x04	OIF	Word	Out-of-IMA frame counter. While ILRCNTL[SES] = 0, increments each time an Out-of-IMA frame anomaly occurs on this link. Initialize to zero at link startup.

39.4.6 Structures in External Memory

The MPC8560's IMA implementation requires supporting queue structures in external memory. These are used for the jitter buffers (on transmission) and the delay compensation buffers (on reception). These structures reside in a 1-Mbyte memory region defined by IMAEXTBASE, and may reside on the local bus, as defined by IMACNTL[DSB].

39.4.6.1 Transmit Queues

Transmit queues are allocated on a per-link basis. They are circular queues of 64-byte buffers, defined by their start and end pointers. For the transmit queue of the timing reference link (TRL), the queue must consist of a minimum of four buffers (although it may consist of five buffers, if consistency of data structures is desired). For the transmit queues of non-TRL links, the queues must consist of five buffers. Queue fill and extract pointers must be initialized by software to the start of the queue; thereafter, these pointers are managed by the MPC8560. The queue pointers must be on a 64-byte aligned boundary.

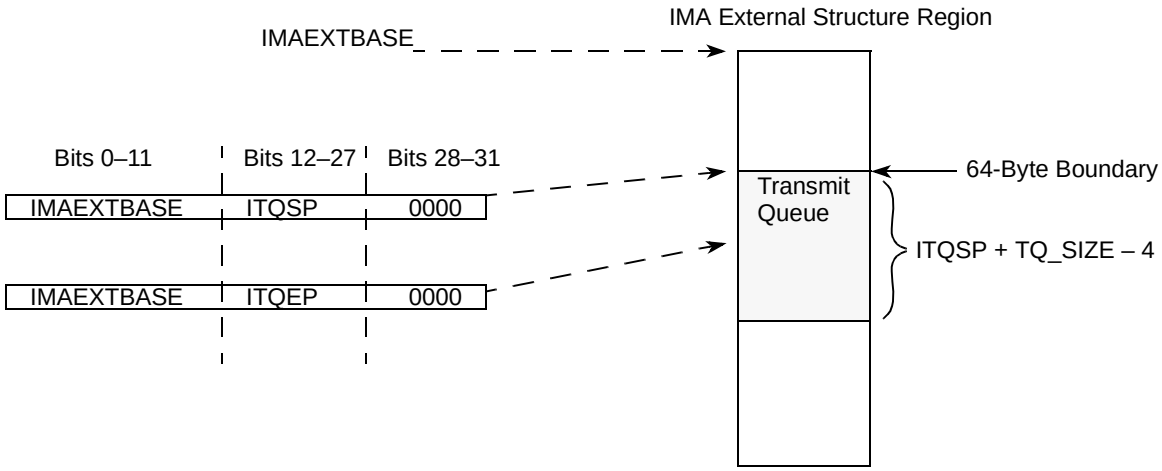


Figure 39-26. IMA Transmit Queue

39.4.6.2 Delay Compensation Buffers (DCB)

Cells received on a link are initially stored in a delay compensation buffer (DCB). DCBs are allocated on a per-link basis. They are of user-definable length, thereby providing a programmable maximum synchronized delay. Note that DCBs of links within a group must all have the same size.

DCBs consist of a circular queue of 64-byte cell buffers. These cell buffers contain the received cell followed by 12 bytes of header/status information.

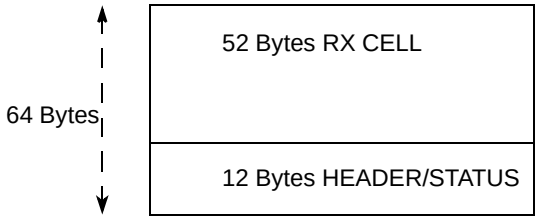


Figure 39-27. Cell Buffer in Delay Compensation Buffer

The length of a DCB is defined by the link's DCBSP and DCBEP parameters. The length of the DCB (defined by (DCBEP – DCBSP) × 16) must be an integer multiple of the IMA frame length in bytes, M × 64. Furthermore, the DCBSP must be aligned on a M × 64-byte boundary. For example, if M = 64, DCBSP must be on a 4KB boundary. To ensure group delay synchronization, the minimum length of the DCB should be 2(M × 64).

The DCB memory area must be initialized to zero at link startup.

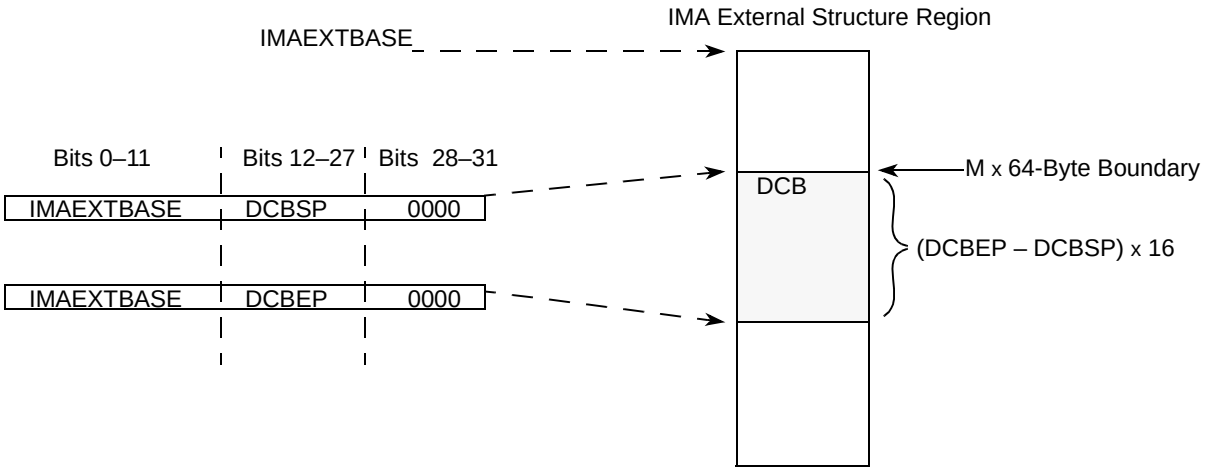


Figure 39-28. IMA Delay Compensation Buffer

39.4.7 IMA Exceptions

One of the four ATM interrupt queues must be dedicated to IMA events. This enables minimum latency in dealing with the IMA state machines, and ensures that the unique format for IMA exceptions are not confused with other exceptions. The IMA interrupt queue is defined in the IMACNTL[INTQ] parameter.

IMA events sent to this queue include only those described in this section. ICP receive events are treated as normal receive events, and should therefore go to the interrupt queue allocated for receive events.

39.4.7.1 IMA Interrupt Queue Entry

The format for the IMA interrupt queue entries is shown in Figure 39-29

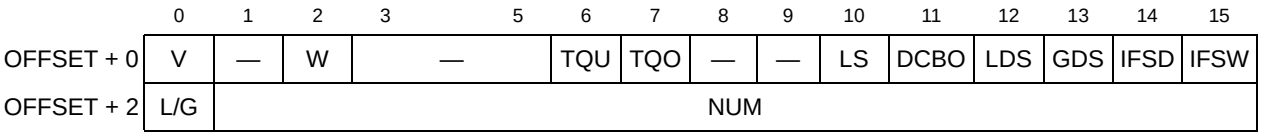


Figure 39-29. IMA Interrupt Queue Entry

Table 39-23 describes the IMA interrupt queue entry bit fields.

Table 39-23. IMA Interrupt Queue Entry Field Descriptions

Offset	Bits	Name	Description
Offset + 0	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	—	Reserved.
	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.
	3–5	—	Reserved
	6	TQU	Transmit queue underrun. Indicates that the corresponding PHY for this link requested a cell for transmission, but the transmit queue was empty.
	7	TQO	Transmit queue overflow. Indicates that the TRL attempted to send a cell to this link's transmit queue, but no space was available.
	8-9	—	Reserved
	10	LS	Link stalled. A link in the round-robin cell extraction process has excessively stalled and been deactivated (that is, switched to filler mode by the MPC8560).
	11	DCBO	Link out of delay synchronization. Set when the link's DCB overflows, indicating that delay synchronization for this link is not possible.
	12	LDS	Link delay synchronized. Set when delay synchronization is achieved for a link that has been added to an existing group.
	13	GDS	Group delay synchronized. Set when a group achieves delay synchronization as part of the group startup procedure.
	14	IFSD	IMA frame synchronization status = defect. Set when the associated link goes into the Loss of IMA Frame Defect state of the Error/Maintenance State Machine.
	15	IFSW	IMA frame synchronization status = working. Set when the associated link goes into the IMA Working state of the Error/Maintenance State Machine.
Offset + 2	0	L/G	Link/group indicator. Indicates whether this interrupt is associated with a link or a group, and thus if the NUM field is a link number or group number. 0 This interrupt is associated with a link. 1 This interrupt is associated with a group.
	1–15	NUM	Link or group number associated with this interrupt.

39.4.7.2 ICP Cell Reception Exceptions

ICP cells are received as AAL0 in the channel defined in RICPCH. Receive interrupts are provided for this channel if enabled in its associated RCT.

39.4.8 APC Programming for IMA

Dynamically adding and dropping links from a group changes the overall bandwidth of the group. The bandwidth of a particular ATM channel is programmed as a percentage of the overall bandwidth, up to 100% in the case of APC pace of 1. In the case of IMA, it is desirable to allow for the dynamic addition and deletion of links without changing the bandwidth of individual ATM channels, so that ATM traffic contracts will not be violated. To do this without requiring updates of the APC parameters of all of the ATM channels, the APC algorithm must be modified to consider the number of links in the group.

To accomplish this for channels which will be used with IMA groups, the APC parameters should be programmed to define the bandwidth of a channel as a percentage of one link of the IMA group. As such, the APC pace can be greater than 100%, in the case that a channel uses more bandwidth than a single link can provide. The APC pace will be scaled automatically by the IMA group transmit parameter TNUMLINKS. After scaling, if the pace required of the group is still greater than 100% of the group bandwidth, the channel will be rescheduled at 100% of the group bandwidth. Refer to the examples in Table 39-24.

Table 39-24. Examples of APC Programming for IMA

Example	Description
1	[The simplest case.] For an IMA group consisting of one 2 Mbps link, with one CBR channel consuming the full bandwidth of that link (that is, 2 Mbps), TNUMLINKS for the group should be programmed to 1 and the APC pace of the channel should be programmed to 1 (PCR = 1, PCR_Fraction = 0).
2	Another 2 Mbps link is added to the IMA group in Example 1. The overall bandwidth of the group is now 4Mbps. However, TNUMLINKS is now 2, and therefore the programmed PCR will be scaled by 2. [Note that the scaling is done automatically internally; the PCR programmed into the channels transmit connection table entry is not modified.] A scaled PCR of 2 indicates that the channel uses 50% of the bandwidth of the group, or 2 Mbps. Comparing to example 1 above, we see that the bandwidth of the channel has not changed, even though the bandwidth of the group has changed.
3	Consider an IMA group consisting of five 2 Mbps links, over which a 6Mbps CBR channel is to be sent. 6Mbps is 300% of what a single 2 Mbps link can provide, so its pace should be programmed as 1/3 (PCR = 0, PCR_Fraction = 85). Scaling the pace by TNUMLINKS results in 5/3 (PCR = 1, PCR_Fraction = 169), which indicates that the channel will use 60% of the bandwidth of the group, or 6Mbps.

Table 39-24. Examples of APC Programming for IMA

Example	Description
4	Assume that one link is dropped from the IMA group in Example 4. The overall bandwidth of the group is now 8Mbps, and TNUMLINKS becomes 4. Therefore, the pace for the 6 Mbps CBR channel described above scales to 4/3 (PCR = 1, PCR_Fraction = 84), indicating that the channel will use 75% of the bandwidth of the group, which is still 6 Mbps.
5	Assume that two links more are dropped from the IMA group in Example 4. The overall bandwidth of the group is now 4 Mbps, and TNUMLINKS becomes 2. Therefore, the pace for the 6 Mbps CBR channel described above scales to 2/3 (PCR = 0, PCR_Fraction = 170). This is less than 1, which is not possible to support, so it is rounded up to 1. The channel originally programmed for 6Mbps now consumes 100% of the 4 Mbps IMA group.

Per the above explanation and examples, it is seen that TNUMLINKS is the only parameter which needs to be modified by software when a link is added or dropped from an IMA group. All other APC parameters need not be modified. [Note, however, that if links are dropped such that the total scheduled bandwidth of the ATM channels is greater than 100% of the IMA group bandwidth, this will result eventually in APC overruns, and should therefore be corrected and/or avoided.

NOTE

Software should ensure that the length of the APC scheduling table is increased if links are added to the IMA group. When incrementing the number of links in an IMA group, the user might exceed the length of the APC scheduling table; if this happens, the ATM channel is mapped outside of the APC scheduling table and the channel stops.

39.4.8.1 Programming for CBR, UBR, VBR, and UBR+

All APC parameters for CBR, UBR, VBR, and UBR+ channels which will be transmitted over IMA groups should be scaled by the intended steady-state value of TNUMLINKS. Their values should be divided by TNUMLINKS, as they will be scaled by TNUMLINKS when the pacing algorithms are performed. The parameters which must be scaled include: PCR, SCR, OOBR, BT, MCR, and MDA.

39.4.8.2 Programming for ABR

ABR channels are a special case, in that they are not programmed as a percentage of the physical line rate as inferred from the period of requests from the PHY layer. Instead, the rate of an ABR channel is programmed as a percentage of an explicitly-provided parameter, the LINE_RATE_ABR, which is programmed in the APC parameter table for each APC. Therefore, when links are added or dropped from an IMA group which carries ABR traffic, the parameter LINE_RATE_ABR must also be scaled to reflect the change in bandwidth of the group. For example, if TNUMLINKS increases from three to four, then LINE_RATE_ABR must also be multiplied by 4/3.

39.4.9 Changing IMA Version

A new CPCR command has been added to the IMA microcode to change the IMA version on-the-fly without software intervention. Use the following procedure:

1. Before issuing the command, the user should initialize COMM_INFO fields in the parameter RAM as described in Figure 39-30.
2. To issue this FCC command, refer to Section 20.4, “Command Set.” Use opcode 1110(0xE).

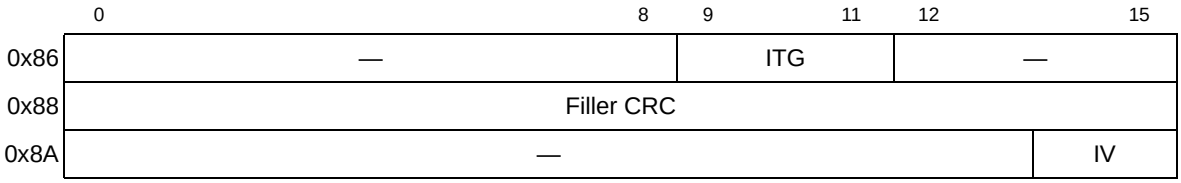


Figure 39-30. COMM_INFO Field

Table 39-25. COMM_INFO Field Descriptions

Offset	Bits	Name	Description
0x86	0–8	—	Reserved, should be cleared.
	9–11	ITG	IMA transmit group. Program to the IMA group number [0–7] multiplied by 16 bytes. For example, if the IMA group number = 3, program ITG to 0x30.
	12–15	—	Reserved, should be cleared.
0x88	0–15	Filler CRC	Filler cell CRC. Set to 0xC602 (for IMA version 1.0) or 0xD902 (for IMA version 1.1)
0x8A	0–13	—	Reserved, should be cleared.
	14–15	IV	IMA version 00 Reserved 01 IMA version 1.0 10 Reserved 11 IMA version 1.1

3. Wait for the CPCR[FLG] to be cleared by the CPM before issuing a new CP command. Refer to Section 20.4, “Command Set.”

39.5 IMA Software Interface and Requirements

39.5.1 Software Model

The MPC8560’s IMA facilitates the implementation of the lower levels of an IMA system. Host software (from here on, “host software” is code running on the G2 core) is responsible for initializing MPC8560 IMA data structures, establishing and tearing down connections, handling of alarms, keeping statistics, and controlling protocol state machines. The MPC8560’s IMA

interfaces to the software-implemented (Layer Management and Plane Management) functions by providing received ICP cells and by interrupts. The software-implemented functions control the MC8560's IMA and the system via the IMA root, group, and link parameters and by providing an ICP cell template to the MPC8560's IMA for ICP cell transmission.

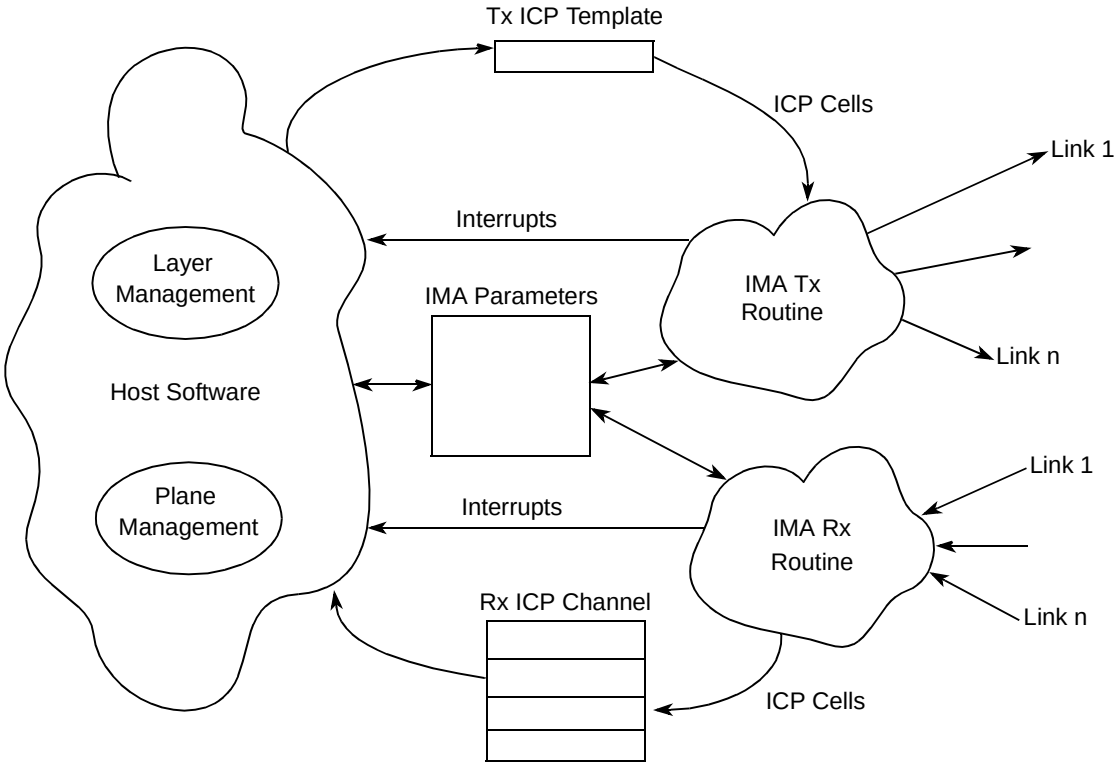


Figure 39-31. MPC8560 IMA Implementation/Software Interaction

39.5.2 Initialization Procedure

1. Program FCC registers/parameters for ATM operation with UTOPIA multi-PHY (excluding APC parameters for IMA PHYs).
2. Program IMA FCC and root parameters.
3. Enable FCC via GFMRx[ENR,ENT].

Aside from IMA state machine control and IMA-specific error events, subsequent interaction with the ATM channels is the same as for non-IMA operation (for example, host commands, RCT/TCT parameters, buffer descriptors, interrupts).

39.5.3 Software Responsibilities

The following functions are the responsibility of the host software which must complement the MPC8560's IMA functionality in order to provide a complete IMA solution.

39.5.3.1 System Definition

- Definition of P(Rx) and P(Tx)—software variables which are used to determine “sufficient links”

39.5.3.2 General Operation

- React to received ICP cells. ICP cells are only received when their SCCI field changes, except the first received ICP cell
- Report any NE and FE timing mode mismatches (ITC vs. CTC) to the Unit Management

39.5.3.3 Receive Link State Machine Control

- Set up unassigned links, awaiting ICP cells
  - Define either as standard multi-PHY or as unassigned IMA link
  - Set up receive channel for ICP OAM cells for this link only
- Create and initialize delay compensation buffer
- Respond to received ICP cells. Extract and validate link/group parameters
  - IMA ID
  - Link ID
  - Link ICP offset
  - IMA frame size (M)
- After establishment of group, change receive channel for ICP OAM cells such that all links in the group point to the same channel
- Control link operation (via ILRCTNL[RXSC]) in coordination with link and group states

39.5.3.4 Receive Group State Machine Control

- Coordinate receive links, using received ICP cells to identify a proposed group
- Establish and validate group parameters in conjunction with the received link states and parameters
  - IMA version
  - IMA ID
  - Group order table
- Enable delay synchronization for the group, and react to its completion
- Control group operation (via IGRCNTL[RXSC]) in coordination with group state
- Signal receive group state via ICP cells of corresponding transmit group

39.5.3.5 Transmit Link State Machine Control

- Define the IMA link
  - Define link parameters (for example, IMA ID, Link ID, IMA frame size (M)) in coordination with IMA group definition
  - Assign ICP offset
  - Create and initialize transmit queue
- Control link operation (via ILTCNTL[TXSC]) in coordination with link and group states

39.5.3.6 Transmit Group State Machine Control

- Define the IMA group(s)
  - Assign IMA ID
  - Assign Link IDs and the transmit group order table
  - Assign TRL
  - Assign IMA frame size (M)
  - Signal group parameters via ICP cells
  - Define ATM pace controller (APC) parameters for this transmit group
- Signal transmit group state via ICP cells
- Negotiate transmit group parameters with the far end
- Check status of links in group to make ‘Sufficient Links’ determination
- Control group operation (via IGTCNTL[TXSC]) in coordination with group state
- Coordinate link state transitions with group state transitions during group startup and link addition

39.5.3.7 Group Symmetry Control

All types of group symmetry (operation and configuration) are supportable. This is facilitated by the ability to independently enable/disable links (via RXPHYEN and TXPHYEN) and to control link and group states (via independent link and group transmit and receive parameters).

39.5.3.8 ICP End-to-End Channel Transmission

- Can send information on end-to-end channel by updating field in Tx ICP.
- No Rx support is provided for end-to-end channel.

39.5.3.9 Link Addition and Slow Recovery (LASR) Procedure

Coordinate addition of links for both receive and transmit, including the following:

- Establishing their parameters
- Checking for defects
- On-the-fly insertion into data structures

39.5.3.10 Failure Alarms

- React to errors signalled in received ICP cells
- React to physical layer errors
- Monitor persistence of errors to determine alarm condition.
- Report receive defects within 2M cells of entering defect state
- Signal upper-layer software

39.5.3.11 Test Pattern Control

- Initiate transmit test patterns in the group's transmit ICP cells (via TXTC and TXTP), and monitor response in the receive ICP cells of the associated receive group
- Respond to test patterns by:
  - Recognizing test pattern activity in the received ICP cells
  - Locating the Tx Test Pattern field in the ICP cell of the test link. (Because the SCCI field changes as part of the test pattern generation by the far end, the cell will necessarily be among the received ICP cells. Locate the latest ICP cell with a LID matching the Tx LID of the test link).
  - Signalling back via the transmit ICP cells of the associated transmit group, by modifying the transmit ICP cell template appropriately

39.5.3.12 Performance Parameter Measurement and Reporting

- Establish timers to correlate errors with time intervals (for example, to determine severely errored seconds (SES) or unusable seconds (UUS))
- Maintain statistics
- Enable/disable receive and transmit event counters according to severely errored seconds (SES) condition via ILRCNTL[SES] and ILTCNTL[SES]

39.5.3.13 SNMP MIBs

Control interface and statistics information should be provided per the SNMP MIB definition for IMA.

39.5.4 IMA Software Procedures

The following procedures must be followed in order to assure the synchronization of changes between the link state machines and the group state machine. Issues to be considered are order of changes to the ICP cell and pointer, group order structure and pointer, IMA PHY assignment structure, and group/link Tx control fields.

39.5.4.1 Transmit ICP Cell Signaling

1. Copy the transmit ICP cell template currently in use (as indicated by TICPPTR) to the ICP cell template area not in use.
2. In the new template, change the ICP cell template fields as appropriate.
3. Verify that the IGTCNTL[ICPC] equals IGTSTATE[ICPCA]. If not, wait until it does.
4. Change TICPPTR to point to the ‘changed/altered’ (see step 1) ICP cell template.
5. Toggle IGTCNTL[ICPC].

39.5.4.2 Receive Link Start-up Procedure

Before ‘activation’ of links and group can take place, ICP cells must be received and processed by the management software. Software must configure all IMA links to ‘group unassigned’ mode. Reception of ICP cells requires that the corresponding PHYs (that is, links) have been enabled and the corresponding connection table entry/entries initialized (see RXPHYEN, IMAPHYEN, and RICPH). In ‘group unassigned’ mode, the first received ICP cell will always be reported to the corresponding channel’s buffer (RICPCH) and subsequently every time there is a change in the SCCI (Status and Control Change Indication) field of the ICP cell.

- Set ILRCNTL[GA] to 0.

Setting GA to zero (group unassigned) allows only ICP cells to be processed, all other cells are dropped. The management software will analyze the ICP cells and program the corresponding IMA Link Receive Table parameters:

- IMA Link ID (ILID)
- Link ICP offset (LICPOS)
- Select link as the Timing Reference Link (only one link can be TRL). ILRCNTL[TRL] = 1.
- Assign the link to a group. ILRCNTL[IGNUM] = x.
- Verify size of frame (M) is the expected value.

The software must have built in knowledge of the mapping between physical links and their corresponding channel number (for example, PHY 0 uses channel 2 (RICPH = 2)). Once a group has been established, the user can have all links in a group report changed ICP cells to a single channel (either by changing RICPCH for all the links to be the same or by clearing the MON_ICP bit):

- ILRCNTL[MON_ICP] = 0. Note, at least 1 link in the group must have this bit set.

39.5.4.3 Group Start-up Procedure

The IMA frame synchronization mechanism (IFSM) is initiated when a link is switched to “group assigned.” However, before that happens, management software must have programmed the group parameters based upon the received/negotiated values in ICP cells:

- IMA ID (RIMAID)
- IMA Version (IMAVER)
- IMA Frame Size (RM)

There are other parameters that don’t depend on ICP information for programmability. Therefore, it is the assumption that they have been initialized prior to the start of the IFSM. Start the IFSM by setting each link in the group to ‘Group Assigned:’

- Set ILRCNTL[GA] to 1

Management software must now wait until each link in the group has achieved IMA frame synchronization. For each link in the group that was ‘group assigned’ an IFSW (IMA Frame Synchronization Working) event is generated. See Section 39.4.7, “IMA Exceptions.”

Having achieved frame synchronization, software can now enable the Group Delay Synchronization (GDS) mechanism (that is, finding link with shortest delay and buffering accordingly via DCB).

- Configure group order table with the ascending link order (round robin distribution) to be used when reconstructing the ATM stream.
- Set the corresponding PHY bit in REF_LINK.
- Set IGRSTATE[GDSS] to 1—to enable GDS.

Once group delay synchronization is achieved, a GDS exception is generated. At this point, ATM stream reconstruction can take place. In order for stream reconstruction to be performed by the MPC8560, the user must switch all links and corresponding group (both directions) to ‘active’ mode (that is, capable of receiving data cells). Set RX to active first, to avoid having the transmitter generate a data stream when the opposite end is not ready:

- Set ILRCNTL[RXSC] to 1—to enable reception of data cells at the link level.
- Set IGRCNTL[RXSC] to 1—to enable reception of data cells at the group level.

- Set ILTCNTL[TXSC] to 1—to enable transmission of data cells at the link level.
- Set IGTCNTL[TXSC] to 1—to enable reception of data cells at the group level.

39.5.4.3.1 As Initiator (TX)

Most of the actions required when a system initiates the establishment of a group with X links involves the exchange of ICP cells between the Near End (Initiator) and the Far End (Responder). In any case, both ends must start with a group of X potential links configured to “filler mode”. One and only one of the links in the group must be designated as TRL.

- Set ILTCNTL[TXSC] to 0—link is in filler mode.
- Set IGTCNTL[TXSC] to 0—group is in filler mode.

The actions at both ends mirror each other. That is, the Near End will initiate the establishment of a group with X links by sending ICP cells to the FE and vise versa. The normal ICP state changes, driven by the state machine software, are (on a per-link basis):

1. Not In Group
2. Unusable
3. Usable
4. Active

Refer to Section 39.5.4.1, “Transmit ICP Cell Signaling,” for details on how to modify and transmit an updated ICP cell.

It is the responsibility of the GSM/LSM (Group/Link State Machine software) to initialize the IMA ID (that is group ID) in the ICP cell template and link ID in the corresponding TX IMA Link Transmit Table Entry (ILTTE):

- Set ILTTE[ILID] = corresponding link ID.
- Set IMA ID accordingly in the ICP cell template.

As an initiator, the NE (‘end’ is relative) must have established a group with X links provisioned.

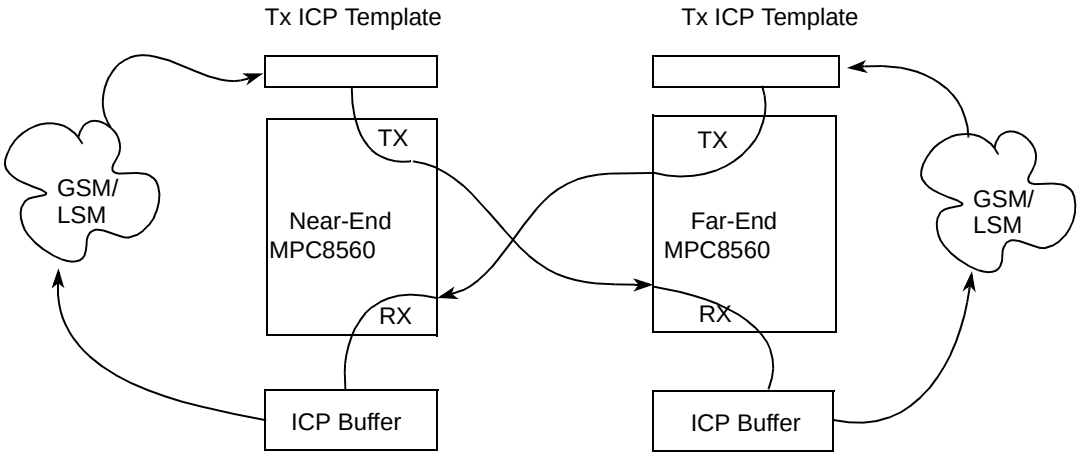


Figure 39-32. Near-End versus Far-End

39.5.4.3.2 As Responder (RX)

The IMA GSM/LSM software will receive ICP cells in the ICP buffer conveying the state of the FE's group and links. Once it has determined that the required minimum number of links are available, it can proceed to enable the 'group delay synchronization (GDS)' mechanism in which the all links in the corresponding group are analyzed to find the one with the shortest propagation delay. The amount of delay tolerated is programmable and is programmed via the setting of the beginning and ending addresses (size) of the delay compensation buffers (DCB). Note that the size of all DCBs must be the same and must be initialized before the GDS mechanism is activated. Activate GDS:

- Set IGRSTATE[GDSS] to 1—to enable GDS. Once GDS is enabled, the user must not alter GDSS.

Once group delay synchronization is achieved, a 'GDS' exception is generated. At this point, ATM stream reconstruction can take place. In order for ATM stream reconstruction to be performed by the MPC8560, the user must switch all links in the group to 'active' mode (that is, capable of receiving data cells), as specified earlier in this section. The initialization of the ATM TX and RX data structures required to generate and receive the ATM stream is out of the scope of this document, however, it is documented in Chapter 35, "ATM Controller."

39.5.4.4 Link Addition Procedure

Adding a link to an existing group requires changes to both the group and link table entries. Aside from the normal exchange of ICP cells by the state machines at the FE and NE, the following steps should be followed. In general, a new link entry is appended to the existing list of link table entries and the required parameters initialized. This has no impact until the corresponding link is enabled via the PHYEN fields in the IMA root table.

39.5.4.4.1 Rx Steps

1. Reset all RX parameters in the new link table entry to zero.
2. Assign corresponding group number for the new link: ILRCNTL[IGNUM] = x.
3. Assign channel number for ICP cell reception: RICPCH = x.
4. Enable desired interrupts: IRINTMSK = x
5. Allow for the reception of ICP cells during the IFSM stage: ILRCNTL[MON_ICP] = 1.
6. Indicate that this link has not yet been assigned to a group: ILRCNTL[GA] = 0.
7. Configure this link/PHY as an IMA link in the IMA Root Table by setting the corresponding bit in IMAPHY. See Table 39-3.
8. Enable the corresponding link/PHY in the IMA Root Table by setting the corresponding bit in RXPHYEN. See Table 39-3.
9. Software receives and accepts ICP cell values (for example, M, LID, etc.).
10. Program expected LID: ILID = x
11. Program the expected ICP offset: LICPOS = x
12. Initialize the DCB pointers accordingly: DCBEP, DCBSP, DCBFP. Note, it is recommended that the DCB be initialized to zero.
13. Configure link to “Loss of IMA Frame” state: ILRSTATE[FSES] = 2.
14. Start the IMA Frame Synchronization Mechanism (IFSM) by assigning this link to the group: ILRCNTL[GA] = 1.
15. Software must now wait for the IFSW (IMA Frame Synchronization Working) event/exception.
16. Now that we have a “frame” synchronized link, we can proceed to allow the link to be “delay” synchronized. Indicate that this is a new link (GDS/reconstruction function) by inverting the current “add-new” bit value: ILRCNTL[ADD_NEW] = x.
17. Formulate new group order table with the new link included (see Section 39.4.4.2.4, “Receive Group Order Tables”).
18. Use the new group order table by inverting the current GOTP value: IGRCNTL[GOTP] = x.
19. The “Stall Threshold” needs to be recalculated. This parameter defines the acceptable tolerance to an emptied DCB condition (stalled link, see LS exception). The recommended new value is: STALL_THR = 2 x RNUMLINKS x (3 + RX_FIFO). See Section 39.4.4.2, “IMA Group Receive Table Entry”: IGRTE[STALL_THR] = x.
20. Start the delay compensation process for this link (in IMA Root Table) by setting the corresponding bit in REF_LINK. See Table 39-3.

21. Software must now wait for the link delay synchronization process to complete. A LDS (Link Delay Synchronized) exception will be generated by the MPC8560 as soon as this happens.
22. It is now safe for the link to receive data, set link to “active” mode: ILRCNTL[RXSC] = 1.

39.5.4.4.2 TX Parameters

1. Reset all TX parameters in the new link table entry to zero.
2. Assign corresponding group number for the new link: ILTCNTL[IGNUM] = x.
3. Enable desired interrupts: ITINTMSK = x.
4. Software formats content of ICP template accordingly. (See Section 39.5.4.1, “Transmit ICP Cell Signaling.”)
5. Program the Link’s ID (LID) in the IMA Link Transmit Table Entry (ILTTE): ILID = x.
6. Program the ICP offset (ILTTE): LICPOS = x.
7. Initialize the Transmit Queue pointers accordingly: ITQSP, ITQEP, ITQFP, and ITQXP.
8. Construct the new TX Group Order Table (includes the added/new link).
9. Point to new TX Group Order Table in the corresponding IMA Group Transmit Table Entry (IGTTE): TGRPORDER = New Table Offset.
10. Configure this link/PHY as an IMA link in the IMA Root Table by setting the corresponding bit in IMAPHY. See Table 39-3.
11. Enable the corresponding link/PHY in the IMA Root Table by setting the corresponding bit in TXPHYEN. See Table 39-3.
12. Software must now wait for the corresponding FE link to go to “active” state (See Section 39.4.4.1.4, “ICP Cell Templates.”). At this point, the link can be configured to “active mode”, capable of sending data cells. Prior to this point, only filler and ICP cells were transmitted. ILTCNTL[TXSC] = 1.
13. Increment the number of links currently in the group (IGTTE): TNUMLINKS + = 1.

39.5.4.5 Link Removal Procedure

Removing a link from an existing group requires changes to both the group and link table entries. Aside from the normal exchange of ICP cells by the state machines at the FE and NE, the following steps should be followed. In general, a link entry is removed from the existing list of link table entries and the required parameters initialized. Note that if only one link is used in a group the software must monitor the TC layer in order to detect that this link has stalled.

39.5.4.5.1 Rx Steps

1. Formulate new group order table with the “dropped” link excluded (see Section 39.4.4.2.4, “Receive Group Order Tables”).
2. Use the new group order table by inverting the current GOTP value: IGRCNTL[GOTP] = x.
3. The “Stall Threshold” needs to be recalculated. This parameter defines the acceptable tolerance to an emptied DCB condition (stalled link; see LS exception). The recommended new value is: STALL_THR = 2 x RNUMLINKS x (3 + RX_FIFO). See section “IMA Group Receive Table Entry”: IGRTE[STALL_THR] = x.
4. Inhibit storing of cells in DCB for “dropped” link (in IMA Root Table): REF_LINK &= ~x (that is, clear the corresponding link bit in the REF_LINK entry).
5. Indicate that the link should be dropped: ILRCNTL[RXSC] = 2.
6. Software should wait (poll) for the MPC8560 to remove the link from the DCB routine. The corresponding bit in the group table’s LINK_DCB entry will be cleared by the MPC8560 (IMA) (this means no more cells are being stored in the DCB), for example, while (LINK_DCB != REF_LINK).
7. Indicate that this link is no longer assigned to a group: ILRCNTL[GA] = 0.
8. Inhibit reception of cells over dropped link in the IMA Root Table: RXPHYEN &= ~x (that is, clear the corresponding link bit in the RXPHYEN entry). Note, you must re-enable (set to 1) the corresponding bit at a later point if you wish to use this link as a non-IMA link.
9. Software should wait (poll) for the MPC8560 to be using the new group order table. This simply ensures that it is safe to modify/re-use the dropped link parameters (that is, the MPC8560 is no longer using the dropped link’s data structures). Do this by making sure the group order pointer points to the new group order table (at this point, no more cells are extracted out of the dropped link’s DCB), for example, while (dcblink != new_pointer). DCBLINK is an entry in the IGRTE.
10. Indicate that this link is no longer an IMA Link in the IMA Root Table: IMAPHY &= ~x (clear the bit). Note, in a symmetrical operation, this step should be the last one performed (after both RX and TX have been disabled). Setting this bit prematurely will halt transmission/reception on the corresponding link.

39.5.4.5.2 TX Parameters

1. Formulate new group order table with the “dropped” link excluded (see Section 39.4.4.1.3, “Transmit Group Order Table”).
2. Point to new TX Group Order Table in the corresponding IMA Group Transmit Table Entry (IGTTE): TGRPORDER = New Table Offset.
3. Decrement the number of links in the group (IGTTE): TNUMLINKS -= 1.

4. Inhibit transmission of cells over dropped link in the IMA Root Table: TXPHYEN &= ~x (that is, clear the corresponding link bit in the TXPHYEN entry). Note, you must reenable (set to 1) the corresponding bit at a later point if you wish to use this link as a non-IMA link.
5. Indicate that this link is no longer an IMA Link in the IMA Root Table: IMAPHY &= ~x (clear the bit). Note, in symmetrical operation, this step should be the last one performed (after both RX and TX have been disabled). Setting this bit prematurely will halt transmission/reception on the corresponding link.

39.5.4.6 Link Receive Deactivation Procedure

The following procedure assumes that the link was part of the IMA group during the group delay synchronization procedure and that an IFSW (IMA Frame Synchronization Working) event has already been received for the link.

1. Formulate new group order table with the “dropped” link excluded (see Section 39.4.4.2.4, “Receive Group Order Tables”).
2. Decrement RNUMLINKS in the group receive table
3. The “Stall Threshold” needs to be recalculated. This parameter defines the acceptable tolerance to an emptied DCB condition (stalled link, see LS exception). The recommended new value is: STALL_THR = 2 x RNUMLINKS x (3 + RX_FIFO). See Section 39.4.4.2, “IMA Group Receive Table Entry”: IGRTE[STALL_THR] = x.
4. Inhibit storing of cells in DCB for “dropped” link (in IMA Root Table): REF_LINK &= ~x (that is, clear the corresponding link bit in the REF_LINK entry).
5. Indicate that the link should be dropped: ILRCNTL[RXSC] = 2.
6. Software should wait (poll) for the MPC8560 to remove the link from the DCB routine. The corresponding bit in the group table’s LINK_DCB entry will be cleared by the MPC8560 (IMA) (this means no more cells are being stored in the DCB), for example, while (LINK_DCB != REF_LINK).
7. Use the new group order table by inverting the current GOTP value: IGRCNTL[GOTP] = x.
8. Set the link to filler mode ILRCNTL[RXSC] = 0
9. Initialize the DCB pointers accordingly: DCBSP=DCBFP, DCBRP=Null.
10. Initialize link DCB in external memory to zero.

39.5.4.7 Link Receive Reactivation Procedure

The following procedure assumes that the link was part of the IMA group during the group delay synchronization procedure and that an IFSW(IMA Frame Synchronization Working) event has already been received for the link.

1. Indicate that this is a new link (GDS/reconstruction function) by inverting the current “add_new” bit value: ILRCNTL[ADD_NEW] = x.
2. Formulate new group order table with the new link included (see Section 39.4.4.2.4, “Receive Group Order Tables”).
3. Use the new group order table by inverting the current GOTP value: IGRCNTL[GOTP] = x.
4. Increment RNUMLINKS in the group receive table.
5. The “Stall Threshold” needs to be recalculated. This parameter defines the acceptable tolerance to an emptied DCB condition (stalled link, see LS exception). The recommended new value is: STALL_THR = 2 x RNUMLINKS x (3 + RX_FIFO). See Section 39.4.4.2, “IMA Group Receive Table Entry”: IGRTE[STALL_THR] = x.
6. Start the delay compensation process for this link (in IMA Root Table) by setting the corresponding bit in REF_LINK. See Table 39-3.
7. Software must now wait for the link delay synchronization process to complete. A LDS (Link Delay Synchronized) exception will be generated by the MPC8560 as soon as this happens.
8. It is now safe for the link to receive data, set link to “active” mode: ILRCNTL[RXSC] = 01.

39.5.4.8 TRL On-the-Fly Change Procedure

Timing Reference Link (TRL) requests (CLAV) drive the distribution of cells to the corresponding link transmit queues. To change the TRL, do the following:

1. Clear TRL bit for the existing TRL: ILTCNTL[TRL] = 0.
2. Set TRL bit for the new TRL: ILTCNTL[TRL] = 1.

Only one link must be selected as “TRL” in a group.

39.5.4.9 Transmit Event Response Procedures

The following TX events may take place when operating in IMA mode. It is recommended that all events be handled via an “exception/interrupt service routine” (ISR) as the response time inherent with interrupt driven events should diminish the negative impact/propagation of such events:

1. TQU (Transmit Queue Underrun)—Indicates that a transmit queue was emptied. This implies that the offending link (PHY) is requesting cells at a faster rate relative to the TRL. If the TRL is out of spec, then you will see multiple links reporting TQUs (because all links in the group will be faster relative to the TRL). The offending link should be removed (see Section 39.5.4.5, “Link Removal Procedure”), check the following: PHY, TX Queue depth (start and end pointers should not be the same), TNUMLINKS is equal (not less or greater) than the number of active links.
2. TQO (Transmit Queue Overflow)—Indicates that a transmit queue was not ready to receive a cell (full). This implies that the offending link (PHY) is requesting cells at a slower rate relative to the TRL. If the TRL is out of spec, then you will see multiple links reporting TQOs (because all links in the group will be slower relative to the TRL). The offending link should be removed (see Section 39.5.4.5, “Link Removal Procedure”), check the following: PHY, TX Queue depth (depth should be the same as other queues), TNUMLINKS is equal (not less or greater) than the number of active links.

39.5.4.10 Receive Event Response Procedures

The following RX events may take place when operating in IMA mode. It is recommended that all events be handled via an “exception/interrupt service routine” (ISR) as the response time inherent with interrupt driven events should diminish the negative impact/propagation of such events:

1. LS (Link Stalled)—The link’s DCB has emptied, either because the PHY is no longer functioning or it is relatively slower than the other links. The offending link should be removed (see Section 39.5.4.5, “Link Removal Procedure”). Make sure the DCB start and end pointers are not the same. If only one link is used in a group the software must monitor the TC layer to detect that this link has stalled. No LS event is reported in the IMA interrupt queue. To re-start an IMA group properly after link(s) have recovered, the user must reset all the MPC8560 IMA-managed parameters in the IMA receive group to zero.
2. DCBO (DCB Overflow)—Indicates that the Delay Compensation Buffer has no more space. The source of the problem can be varied: 1) the offending PHY is sending at a faster rate (out of spec.), 2) the propagation delay of one of the other links in the group is greater than anticipated (need to make DCB larger) or, 3) the size of the offending link was programmed incorrectly, that is, smaller (the size of all DCBs in the group should be the same). The offending link should be removed (see Section 39.5.4.5, “Link Removal Procedure”). This exception corresponds to the LODS (Link Out of Delay

Synchronization Defect) and should be reported to the FE (via the ICP cell, RxDefect = LODS) of the problem. The MPC8560 will continue to report this exception if the condition persists (unless the event is masked).

3. LDS (Link Delay Synchronized)—Indicates that the new/added link (to an existing group) has achieved synchronization (link delay). The link is capable of receiving data at this point, (see Section 39.5.4.4, “Link Addition Procedure”).
4. GDS (Group Delay Synchronized)—Indicates that the new group has achieved synchronization (all links in the group). The group is capable of receiving data at this point.
5. IFSD (Link (IMA) Frame Synchronization Defect)—Indicates that the link has lost synchronization (for example, unexpected IFSN value for a number (GAMMA + 2) of consecutive frames). If this condition persists, the link should be removed. The threshold that would trigger the removal of the link is system specific. The IFSD event corresponds to the “Loss of IMA Frame” (LIF) state and the software should react to this by informing the FE (via the ICP cell, RxDefect = LIF) of the problem. If the condition persists, IFSD events will continue to be generated every GAMMA+2 frames. The MPC8560 maintains counters (if enabled) to track the overall “quality” of a particular link: see OIF (Out of IMA Frame) and ICPVIOL (ICP Violation) in Section 39.4.5.3, “IMA Link Receive Statistics Table.” If the link returns to working state, the MPC8560 will automatically perform Link Delay Synchronization and generate an LDS event upon achieving synchronization. The software can use one of the MPC8560 timers as a time stamp when handling IFSD events and check if the “persistence” time threshold has been crossed when subsequent events take place. The LIF state time stamp should be reset when the IFSW event is reported.
6. IFSW (Link (IMA) Frame Synchronization Working)—Indicates that the link is has achieved frame synchronization. This event is reported when a link has been assigned to a group (start-up or link addition) or it was previously assigned and working, but had a defect (see IFSD event). If going from defect to working, the software should update the RxDefect field of the ICP. Again, a time stamp (timer) can be used to measure the “persistence” time.

39.5.4.11 Test Pattern Procedure

Test patterns are used verify “connectivity” of a link in a particular group. The NE will request, via an ICP cell, that a test pattern be looped back to the FE over all links currently in the group. For this test, the FE will look only at ICP cells received over the link being tested (LID = x) and upon receiving an ICP cell “echo” the pattern back to the NE over all the links in the group. The major task of initiating and verifying the pattern on all the links in the group falls on the software. The MPC8560 will simply transmit and receive the modified/changed ICP cells. If the software detects that the pattern was “echoed” by the FE, then connectivity is verified.

39.5.4.11.1 As Initiator (NE)

Alter the (unused) ICP cell template to the “Test Link” command to the FE:

1. Tx Test Control = Set to “Active” and select LID (link) to be tested.
2. Tx Pattern = X (0-255).
3. Increment SCCI.
4. Make this the active ICP template (see Section 39.5.4.1, “Transmit ICP Cell Signaling”).
5. Repeat steps 1-4 until expected pattern is received in any of the incoming ICP cells (alternate ICP templates must be used).

After sending the request to perform the test (pattern) to the FE, the NE software must wait for the TX test pattern to be echoed back/received in the RX Pattern field of the ICP cell (any of the active links in the group can be monitored for the RX Pattern). After verifying connectivity, the test can be de-activated (set Tx Test Control = “Inactive” and repeat steps 3 and 4 above).

39.5.4.11.2 As Responder (FE)

When the FE receives a request to perform the “pattern” test on any of the active links, it must monitor the selected link (see Tx LID of Tx Test Control) for reception of an ICP cell. Upon receiving an ICP cell (remember, the other end is incrementing the SCCI field), the software should copy the Tx Pattern to the Rx Pattern field of the ICP Cell and transmit the ICP cell. Alter the ICP cell:

1. Rx Pattern = Tx Pattern (of received ICP Cell).
2. Increment SCCI
3. Make this the active ICP template (see Section 39.5.4.1, “Transmit ICP Cell Signaling”).
4. Repeat steps 1–3 until the test is inactive.

There are two ways to monitor the link under test. The first method is to simply look for the expected LID in the received ICP cells (ICP cell buffer). For this to happen, all links in the group must have MON_ICP bit set:

1. Monitor link for changes in SCCI: ILRCNTL[MON_ICP] = 1.

Now, since changed ICP cells are passed to the ICP buffer only when the SCCI field changes, then it may take some time for the link under test to pass on an ICP cell. Again, ICP cells will be passed on to the buffer for the first link encountered in which a change (SCCI) is detected, it may or may not be the link under test. An alternate method is to monitor only the link under test:

1. Don’t Monitor link for changes in SCCI for all links except link under test: ILRCNTL[MON_ICP] = 0 (alter the corresponding Link Table Entries).
2. Monitor link under test for changes in SCCI: ILRCNTL[MON_ICP] = 1.

This will allow only changed ICP cells for the link under test to be passed on to the ICP Cell buffer.

39.5.4.12 End-to-End Channel Signalling Procedure

39.5.4.12.1 Transmit

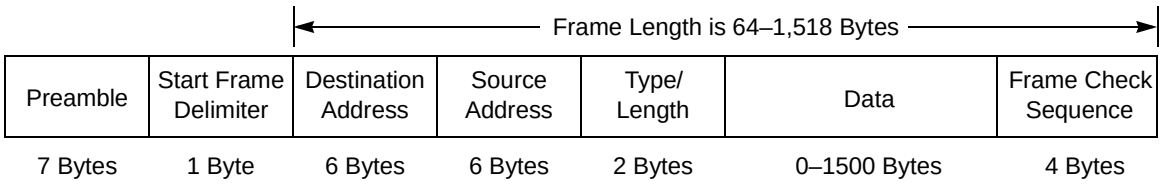
The end-to-end channel signalling field can be used by software to signal to the far end. Because the end-to-end channel is not covered by the SCCI field and there are no standard requirements for the minimum number of frames between changes of the end-to-end channel field, it is not necessary to follow the procedure for ICP cell signalling in order to do this; instead, the end-to-end channel field of the ICP template currently in use can be directly updated. However, if a minimum number of frames between changes is desired, then the procedure for ICP cell signalling can be used, skipping the step in which the SCCI is updated.

39.5.4.12.2 Receive

No special facility is included for reception of the end-to-end channel. The end-to-end channel field is part of the received ICP cells, so its information can be read by software from those cells. However, ICP cells are only received when the SCCI field changes, so changes in the end-to-end channel will not be received until the SCCI field also changes (when there is a change in the status and/or control of the far end).

Chapter 40
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 40-1.



Note: The lsb of each octet is transmitted first.

Figure 40-1. 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.

40.1 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- and 100-Mbps Ethernet are shown in Table 40-1. 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 40-1. Basic Ethernet Timing Specifications

Parameter	Size in Bits	Timing 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

40.2 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 40-2 shows a block diagram of the FCC Ethernet control logic.

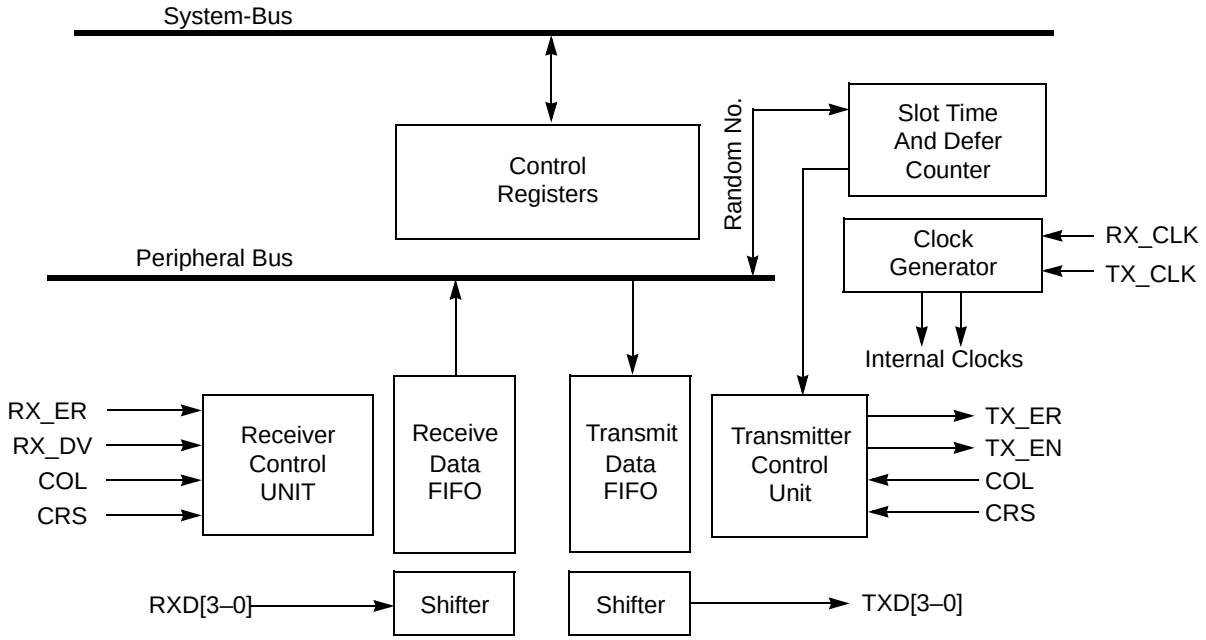


Figure 40-2. Ethernet Block Diagram

40.3 Features

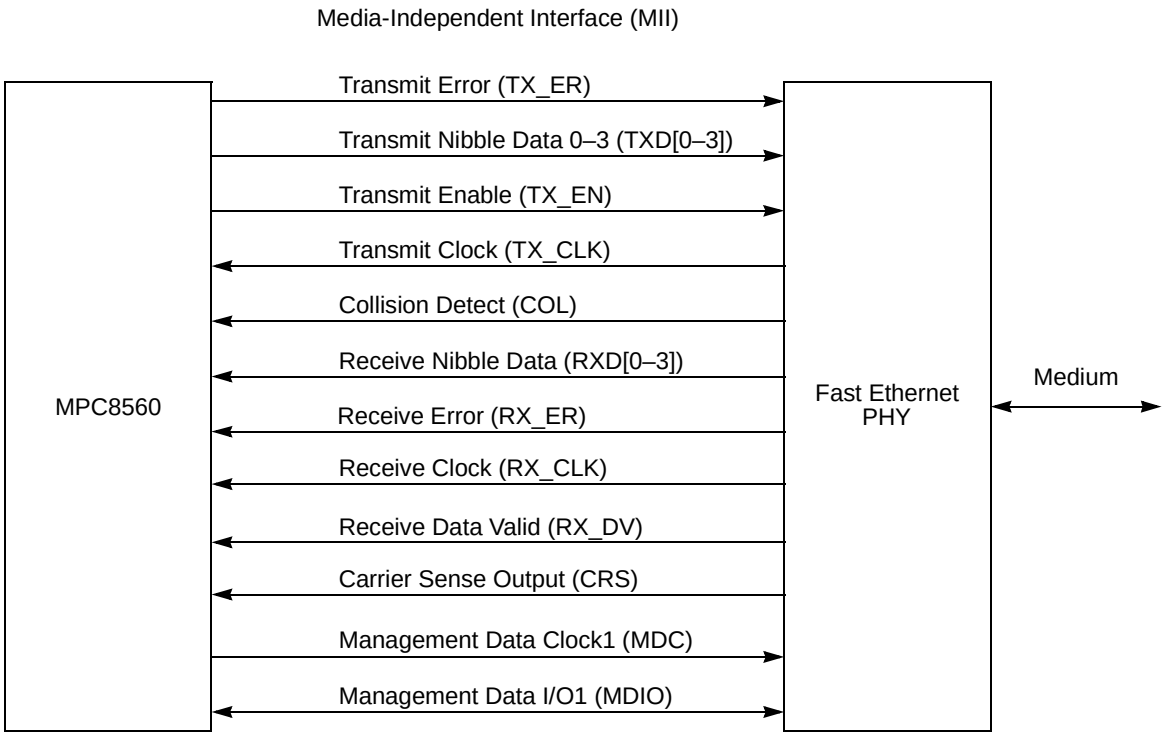
The following is a list of CPM fast Ethernet key features:

- 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)

40.4 Connecting the MPC8560 to Fast Ethernet

Figure 40-3 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 40-3. Connecting the MPC8560 to Ethernet

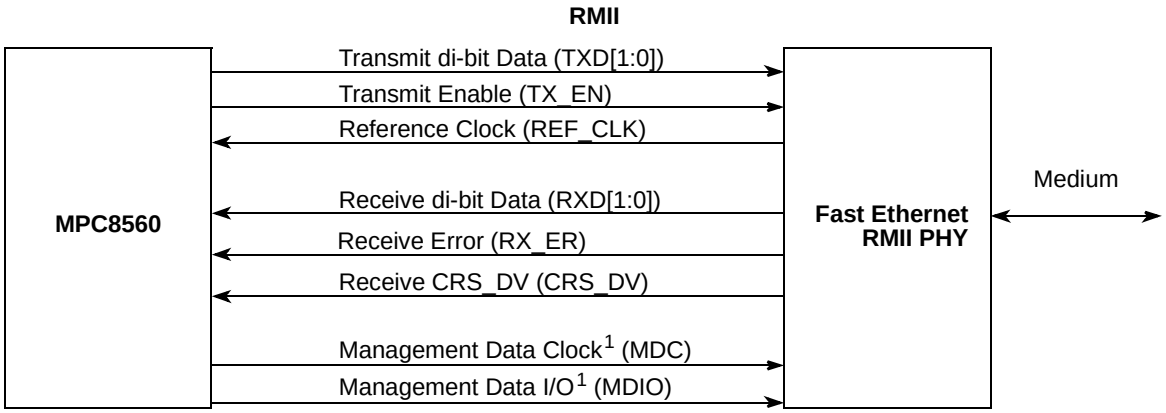
Each FCC has 18 signals, defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY. The two management signals (MDC and MDIO) required by the MII should be implemented separately using the parallel I/O.

The MPC8560 has additional signals for interfacing with an optional external content-addressable memory (CAM), which are described in Section 40.8, “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.

40.4.1 Connecting the MPC8560 to Ethernet (RMII)

Figure 40-4 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 40-4. Connecting the MPC8560 to Ethernet (RMII)

40.5 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 40-1. 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 40-3 on page 40-10.

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.

40.6 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 40.13, "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 40.8, “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.

40.7 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 40-2 shows the flow-control frame structure.

Table 40-2. 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 2 2 x slot time 3 or 4 4 x slot time ……
42	Reserved	—	
4	FCS		Frame check sequence (CRC)

When flow-control mode is enabled (FPSMR*x*[FCE] = 1) 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.

40.8 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 40.13, “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).

40.9 Ethernet Parameter RAM

For Ethernet mode, the protocol-specific area of the FCC parameter RAM is mapped as in Table 40-3.

Table 40-3. Ethernet-Specific Parameter RAM

Offset[1]	Name[2]	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**[3]	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**[3]	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**[3]	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 40.14, “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 40-3. Ethernet-Specific Parameter RAM (continued)

Offset[1]	Name[2]	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 40.14, “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

Table 40-3. Ethernet-Specific Parameter RAM (continued)

Offset[1]	Name[2]	Width	Description
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.
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** [3]	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** [3]	Word	(RMON mode only) The best estimate of the total number of collisions on this Ethernet segment.
0xC8	**BROC** [3]	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** [3]	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** [3]	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** [3]	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** [3]	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** [3]	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.

Table 40-3. Ethernet-Specific Parameter RAM (continued)

Offset[1]	Name[2]	Width	Description
0xE0	**P64C** [3]	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** [3]	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).
0xE8	**P128C** [3]	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** [3]	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** [3]	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** [3]	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 **Boldfaced** entries must be initialized by the user.
3 **32-bit (modulo 232) counters maintained by the CP; cleared by the user while the channel is disabled.**

40.10 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 FDSR as described in Section 34.5, “FCC Data Synchronization Registers (FDSRx),” with FDSR[SYN2] = 0xD5 and FDSR[SYN1] = 0x55.

40.11 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 40-4.

Table 40-4. 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 40-5.

Table 40-5. 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.

40.12 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 40-6.

Table 40-6. 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 40-6. 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

40.13 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 40-5 is a flowchart for address recognition on received frames.

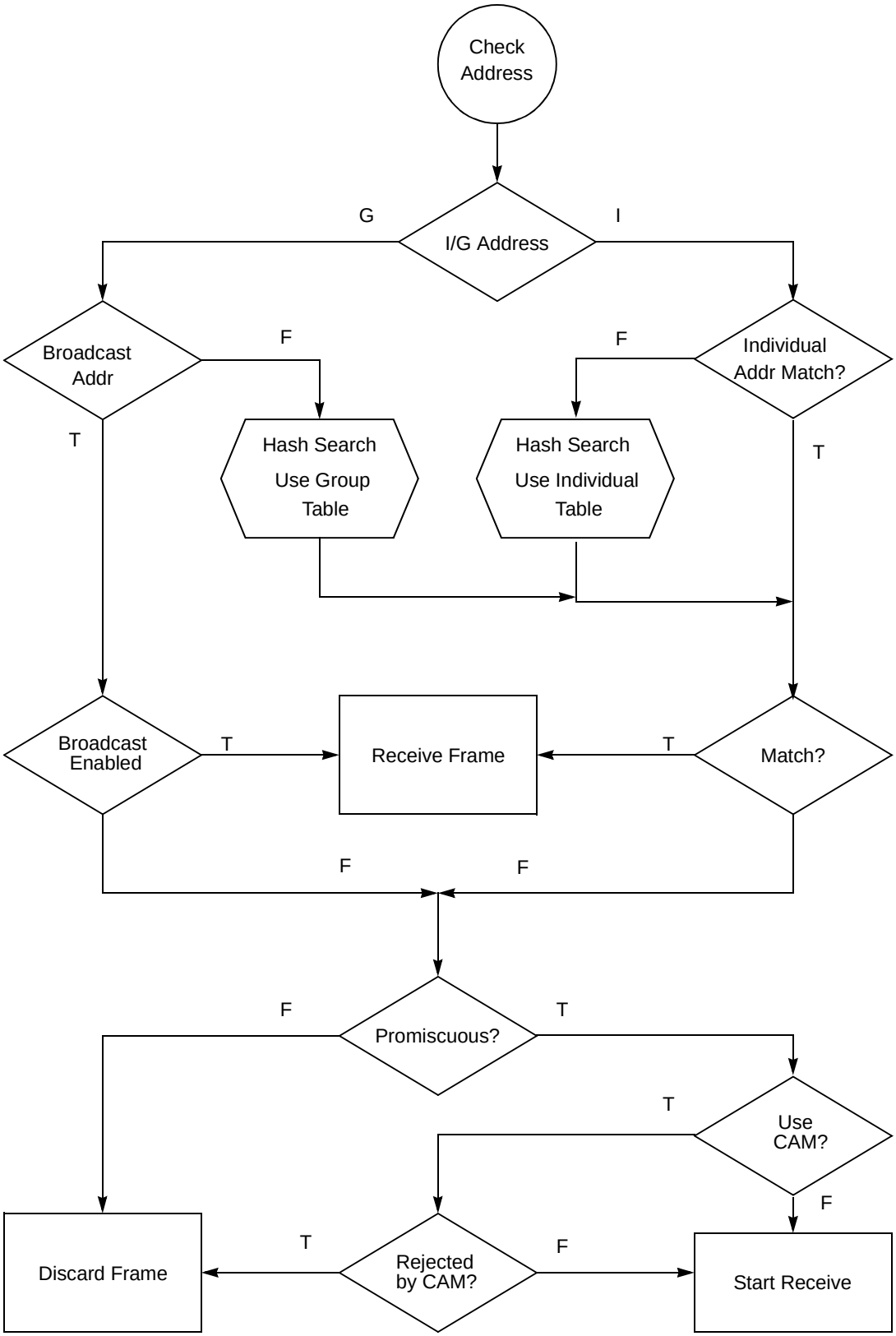


Figure 40-5. Ethernet Address Recognition Flowchart

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.

40.14 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 40.8, “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.

40.15 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

40.16 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.

40.17 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.

40.18 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 40-7.

Table 40-7. 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 40-8.

Table 40-8. 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 40-8. Reception Errors (continued)

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.

40.18.1 FCC Ethernet Mode Register (FPSMR)

The MPC8560 supports 10/100 Mbps Ethernet through an 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 40-6.

	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 40-6. FCC Ethernet Mode Register (FPSMR*x*)

Table 40-9 describes FPSMR fields.

Table 40-9. 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.

Table 40-9. FPSMR Ethernet Field Descriptions (continued)

Bits	Name	Description
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.
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

40.18.2 Ethernet Event Register (FCCE)/Mask Register (FCCM)

The FCCE, shown in Figure 40-7, 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)

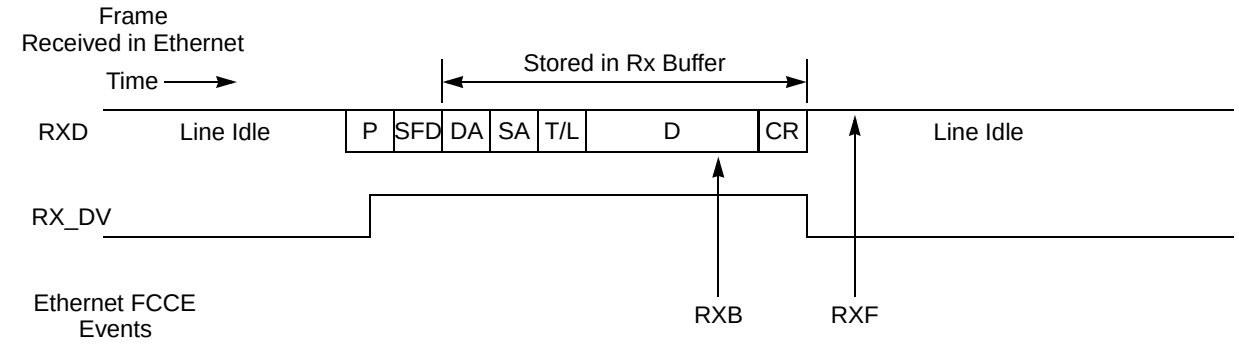
Figure 40-7. Ethernet Event Register (FCCE)/Mask Register (FCCM)

Table 40-10 describes FCCE/FCCM fields.

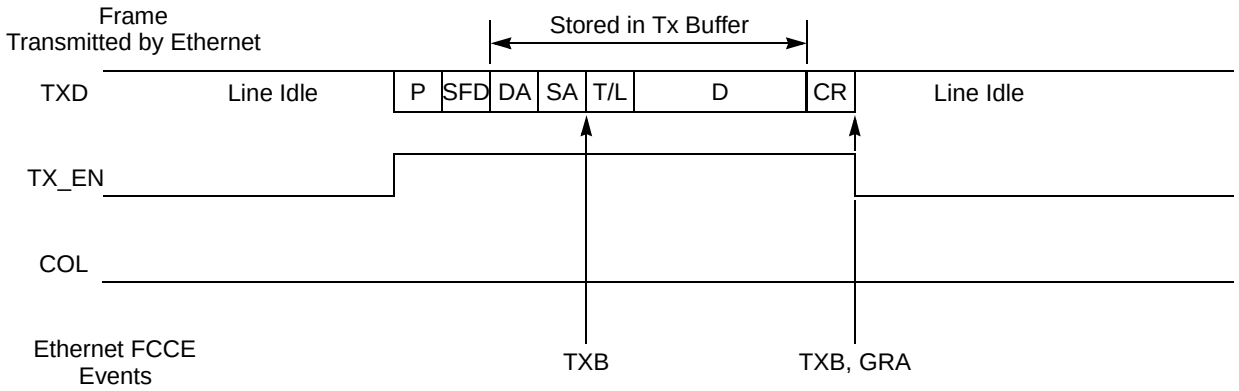
Table 40-10. FCCE/FCCM Field Descriptions

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 40-8 shows interrupts that can be generated in the Ethernet protocol.



Notes:
1. RXB event assumes receive buffers are 64 bytes each.
2. The RXF interrupt may occur later than RX_DV due to receive FIFO latency.



Notes:
1. TXB events assume the frame required two transmit buffers.
2. The GRA event assumes a graceful stop transmit command was issued during frame transmission.

Legend:
P = Preamble, SFD = Start frame delimiter, DA and SA = Destination/Source address,
T/L = Type/Length, D = Data, CR = CRC bytes

Figure 40-8. 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.

40.19 Ethernet RxBDs

The Ethernet controller uses the RxBD to report information about the received data for each buffer. Figure 40-9 shows the FCC Ethernet RxBD format.

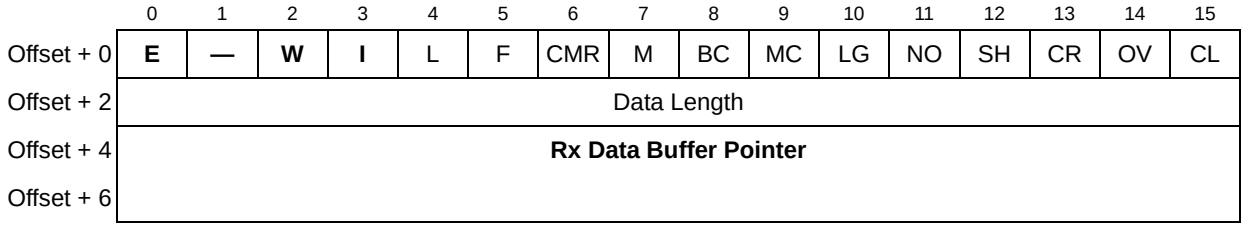


Figure 40-9. Fast Ethernet Receive Buffer (RxBD)

Table 40-11 describes Ethernet RxBD fields.

Table 40-11. RxBD Field Descriptions

Bits	Name [1]	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).

Table 40-11. RxBD Field Descriptions (continued)

Bits	Name [1]	Description
8	BC	Broadcast address. Valid only for the last buffer in a frame (RxBD[L] = 1). The received frame address is the broadcast address.
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].

1 **Boldfaced** entries must be initialized by the user.

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 40-10 shows how RxBDs are used during Ethernet reception.

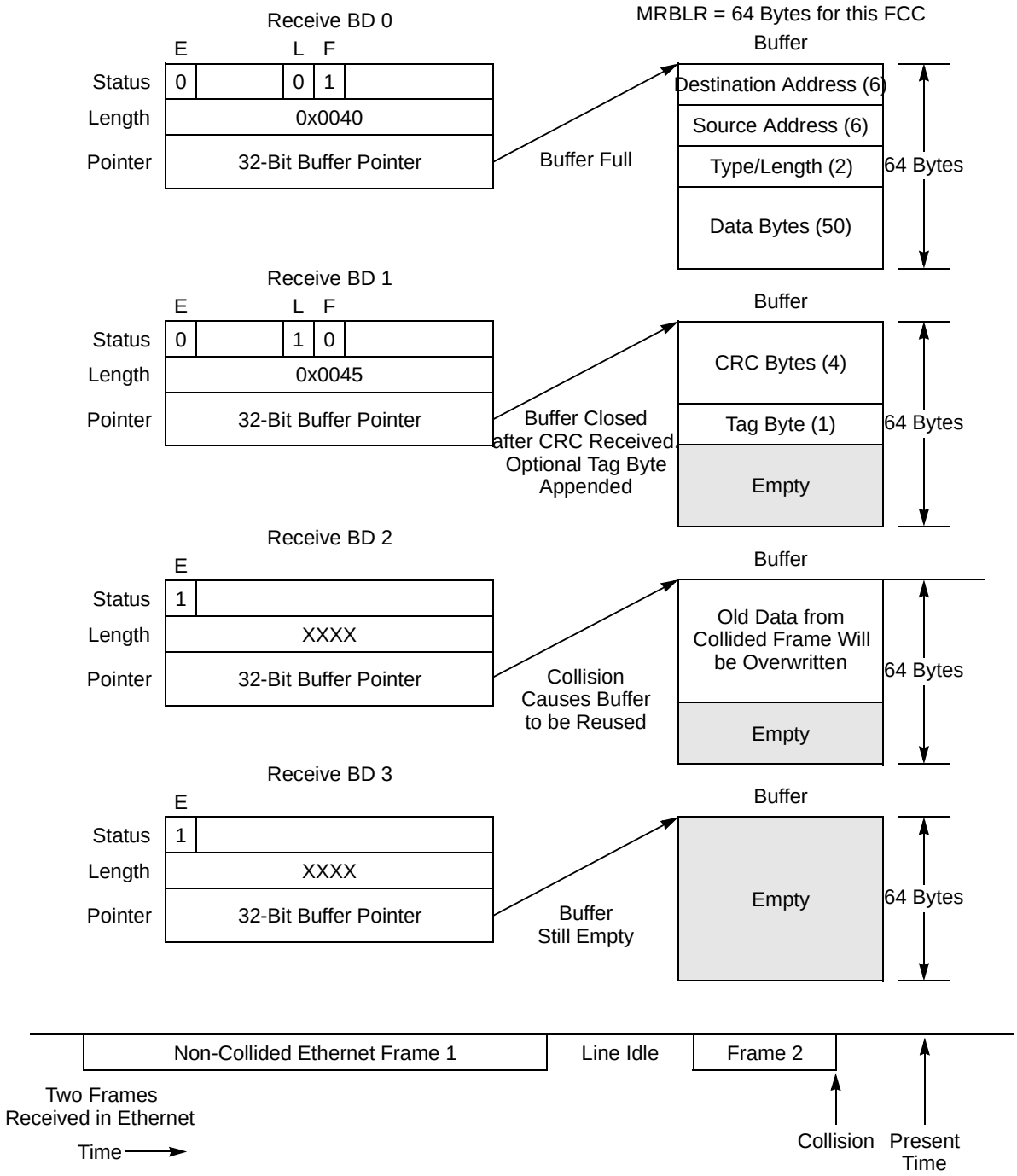


Figure 40-10. Ethernet Receiving Using RxBDs

40.20 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 40-11 shows the FCC Ethernet TxBD format.

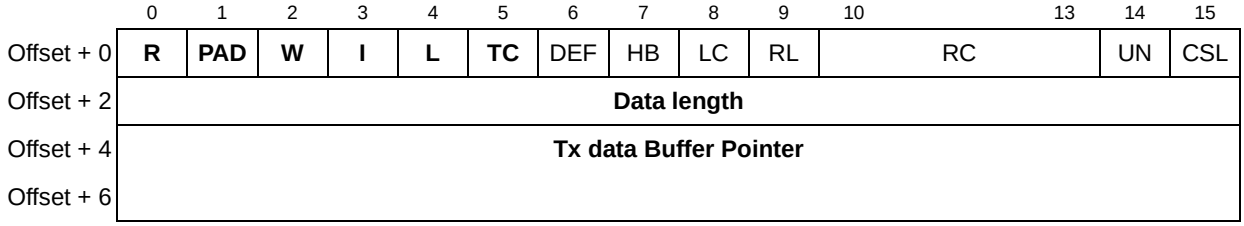


Figure 40-11. Fast Ethernet Transmit Buffer (TxBD)

Table 40-12 describes Ethernet TxBD fields.

Table 40-12. Ethernet TxBD Field Definitions

Bits	Name[1]	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.

Table 40-12. Ethernet TxBD Field Definitions (continued)

Bits	Name[1]	Description
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.
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.

1 **Boldfaced** entries must be initialized by the user.

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 41
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 41-1. 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 always used in point-to-point signaling links. LAPD further divides its 16-bit address into different fields to specify various access points within 1 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 on the protocol using the frame. Data is transmitted in the data field, which can vary in length depending on 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.

41.1 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)

41.2 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 41-1 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 41-1. 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.

41.3 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.

41.4 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 41-1.

Table 41-1. FCC HDLC-Specific Parameter RAM Memory Map

Offset[1]	Name[2]	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**[3]	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, $\overline{\text{CD}}$ 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 41-2.
0x60	**HADDR1**	Hword	
0x62	**HADDR2**	Hword	
0x64	**HADDR3**	Hword	
0x66	**HADDR4**	Hword	

Table 41-1. FCC HDLC-Specific Parameter RAM Memory Map (continued)

Offset[1]	Name[2]	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 **Boldfaced** entries must be initialized by the user.

3 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 41-2 shows an example of using HMASK and HADDR[1–4].

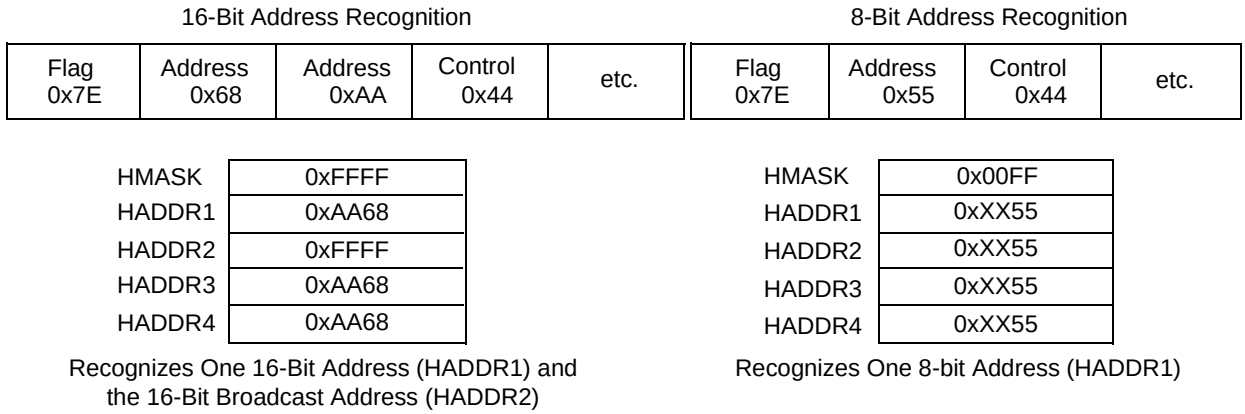


Figure 41-2. HDLC Address Recognition Example

41.5 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.

41.5.1 HDLC Command Set

The transmit and receive commands are issued to the CPCR; see Section 20.4, "Command Set."

Table 41-2 describes the transmit commands that apply to the HDLC controller.

Table 41-2. 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 41-3 describes the receive commands that apply to the HDLC controller.

Table 41-3. 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.

41.5.2 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 41-4 describes HDLC transmission errors, which are reported through the TxBD.

Table 41-4. 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 41-5 describes HDLC reception errors, which are reported through the RxBD.

Table 41-5. 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

Error	Description
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.

41.6 HDLC Mode Register (FPSMR)

When an FCC is configured for HDLC mode, the FPSMR is used as the HDLC mode register, shown in Figure 41-3.

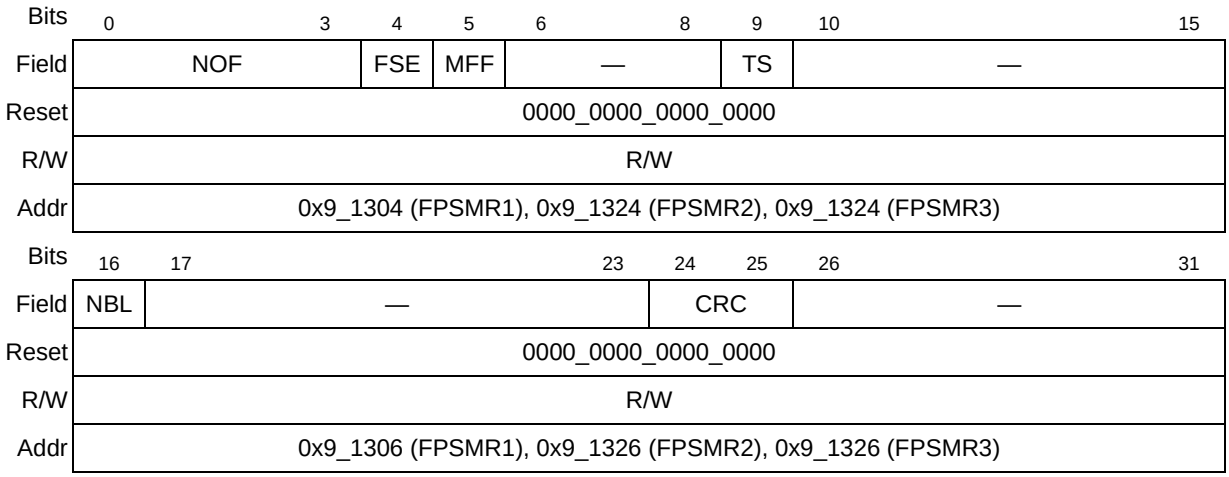


Figure 41-3. HDLC Mode Register (FPSMR)

The FPSMR fields are described in Table 41-6.

Table 41-6. 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 41-6. 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), $\overline{\text{RTS}}$ 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), $\overline{\text{RTS}}$ 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

41.7 HDLC Receive Buffer Descriptor (RxBD)

The HDLC controller uses the RxBD to report on data received for each buffer. Figure 41-4 shows an example of the RxBD process.

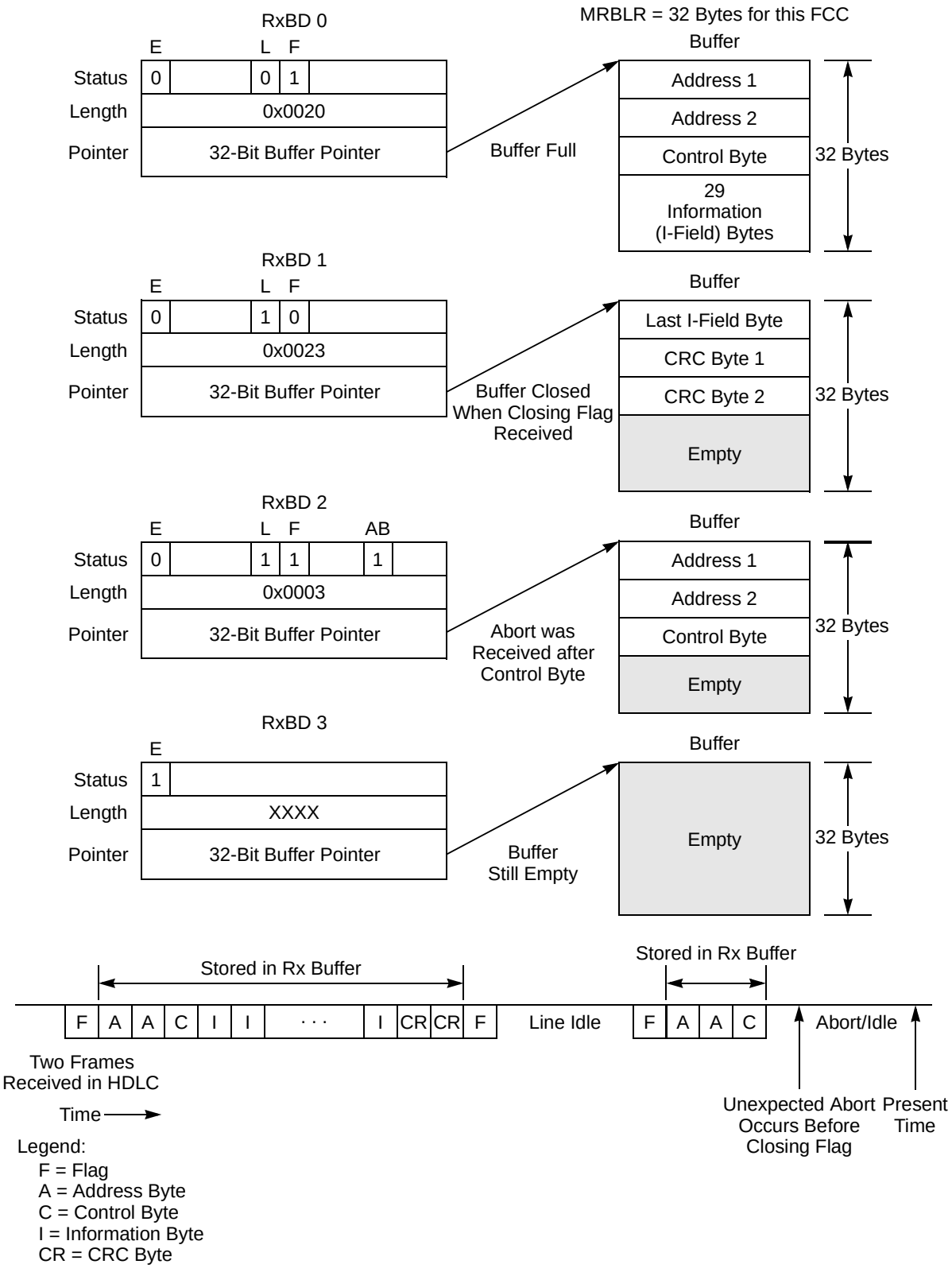


Figure 41-4. FCC HDLC Receiving Using RxBDs

Figure 41-5 shows the FCC HDLC RxBD.

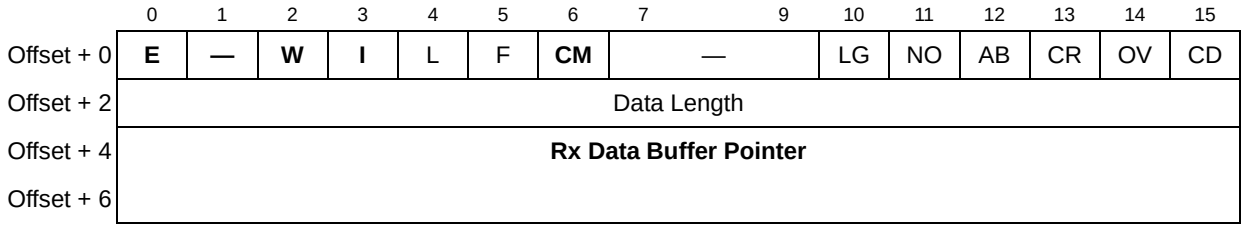


Figure 41-5. FCC HDLC Receive Buffer Descriptor (RxBD)

Table 41-7 describes RxBD fields.

Table 41-7. RxBD Field Descriptions

Bits	Name[1]	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 2 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

Table 41-7. RxBD Field Descriptions (continued)

Bits	Name[1]	Description
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.
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 7 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.

[1] **Boldfaced** entries must be initialized by the user.

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 41-6).

41.8 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 41-6 shows the FCC HDLC TxBD.

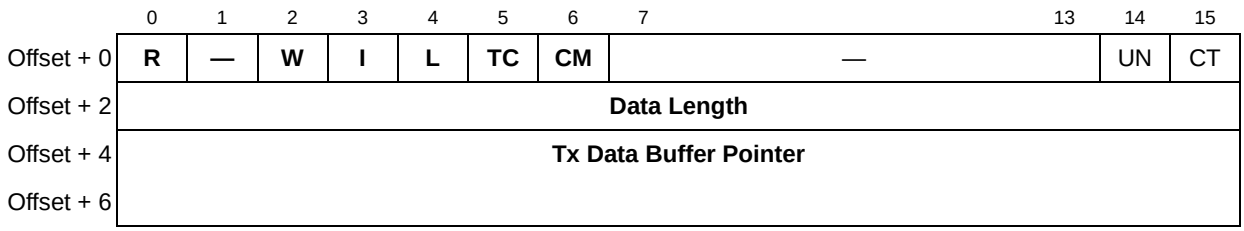


Figure 41-6. FCC HDLC Transmit Buffer Descriptor (TxBD)

Table 41-8 describes HDLC TxBD fields.

Table 41-8. HDLC TxBD Field Descriptions

Bits	Name[1]	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.

[1] **Boldfaced** entries must be initialized by the user.

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.

41.9 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 41-7 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 41-7. HDLC Event Register (FCCE)/Mask Register (FCCM)

Table 41-9 describes FCCE/FCCM fields.

Table 41-9. 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.
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 41.10, “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 41.10, “FCC Status Register (FCCS).”
24–31	—	Reserved, should be cleared

Figure 41-8 shows interrupts that can be generated in the HDLC protocol.

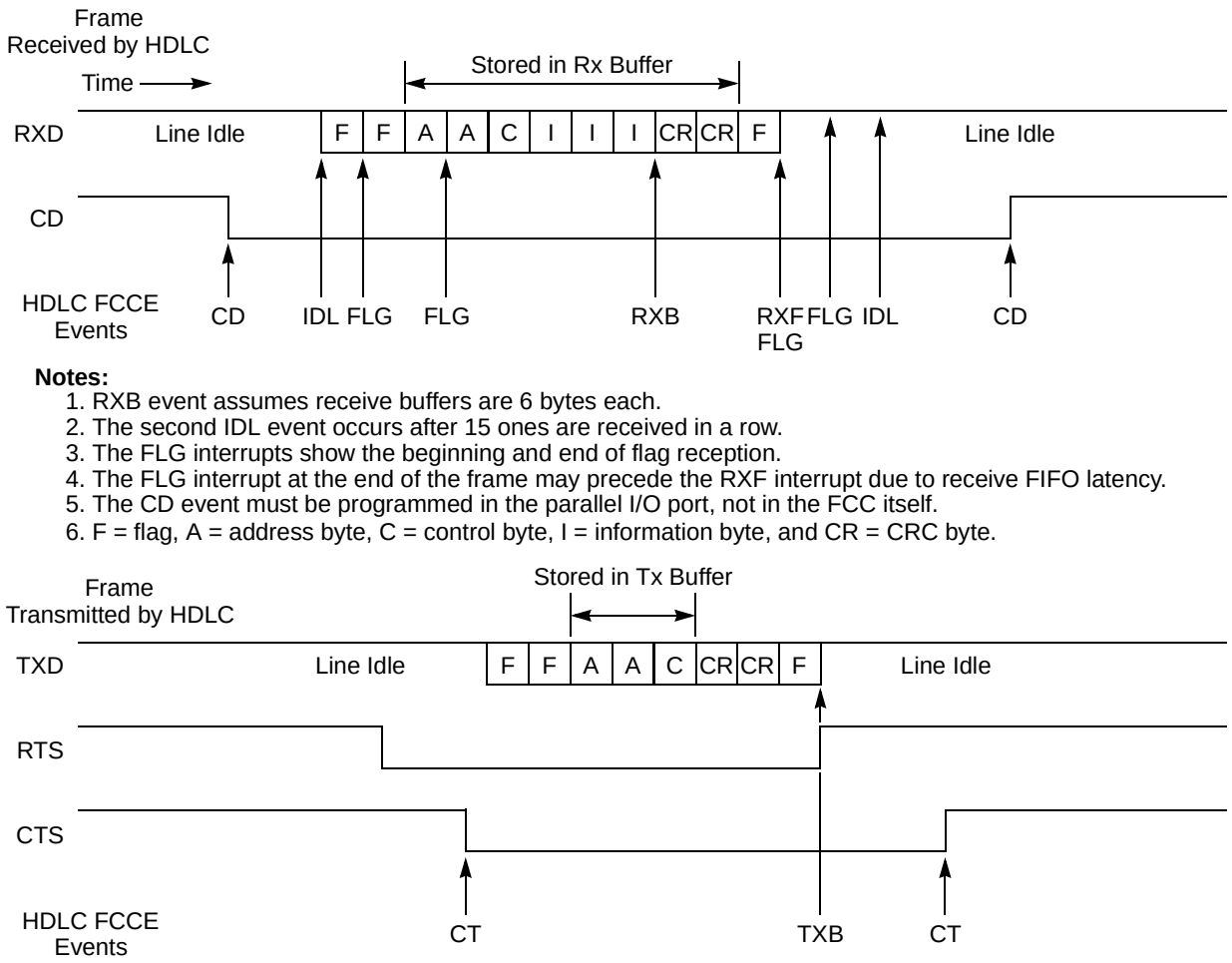


Figure 41-8. HDLC Interrupt Event Example

41.10 FCC Status Register (FCCS)

The FCCS register, shown in Figure 41-9, 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 45, “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 41-9. FCC Status Register (FCCS)

Table 41-10 describes FCCS bits.

Table 41-10. 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 42
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.

42.1 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

42.2 Transparent Channel Operation

The transparent transmitter and receiver operates in the same way as the HDLC controller of the FCC (see Chapter 41, "FCC HDLC Controller") except in the following ways:

1. The FPSMR does not affect the transparent controller, only the GFMR does.
2. In Table 41-1., 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 42.3, "Achieving Synchronization in Transparent Mode."

42.3 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

42.3.1 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 42-1.

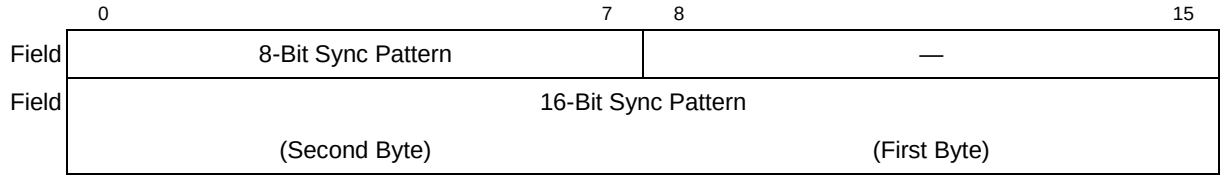


Figure 42-1. In-Line Synchronization Pattern

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.

42.3.2 External Synchronization Signals

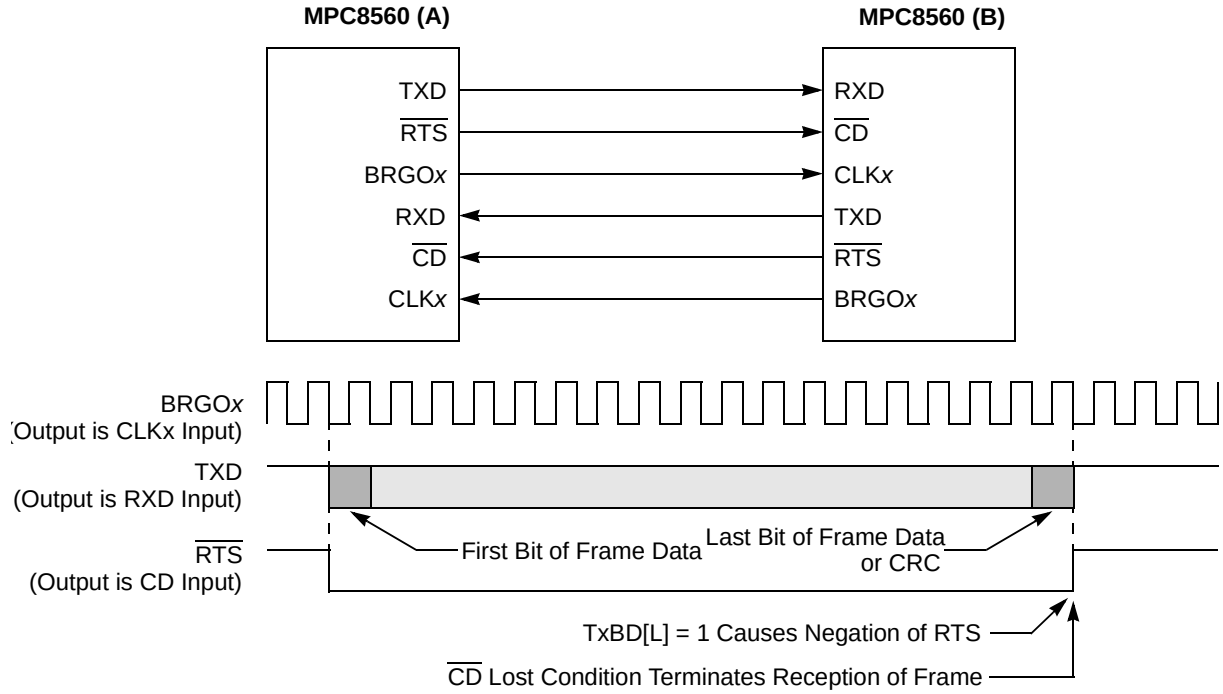
If GFMR[SYNL] = 00, an external signal is used to begin the sequence. $\overline{\text{CTS}}$ is used for the transmitter and $\overline{\text{CD}}$ is used for the receiver; these signals share the following sampling options.

- 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.”

42.3.3 Transparent Synchronization Example

Figure 42-2 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 42-2. 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 43
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 43-1, 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.

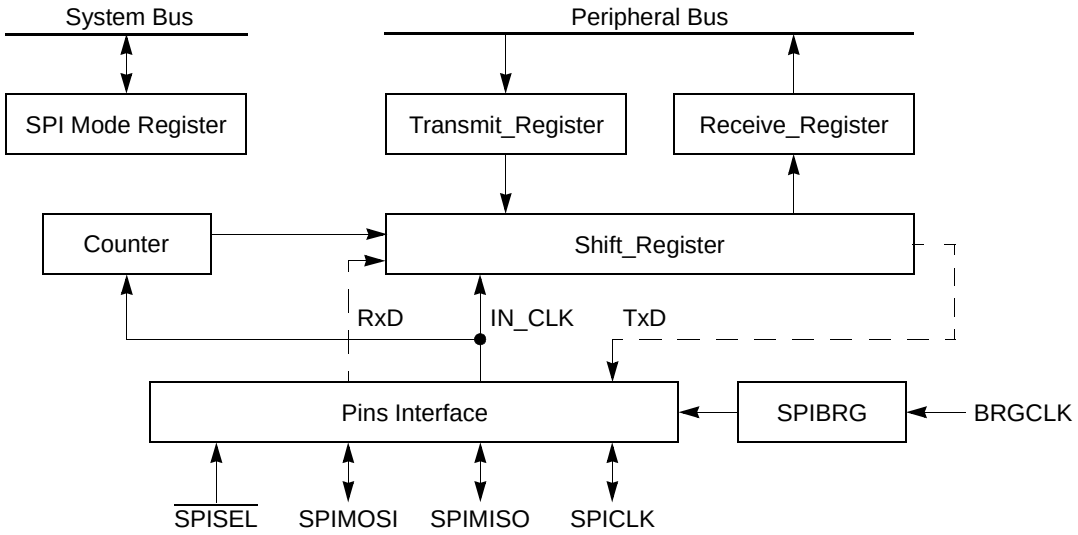


Figure 43-1. SPI Block Diagram

43.1 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 MHz 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

43.2 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 signal (SPIMISO) acts as an input for master devices and as an output for slave devices. Conversely, the master-out slave-in signal (SPIMOSI) 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 45, “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*].

43.3 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.

43.3.1 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 43-2. 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).

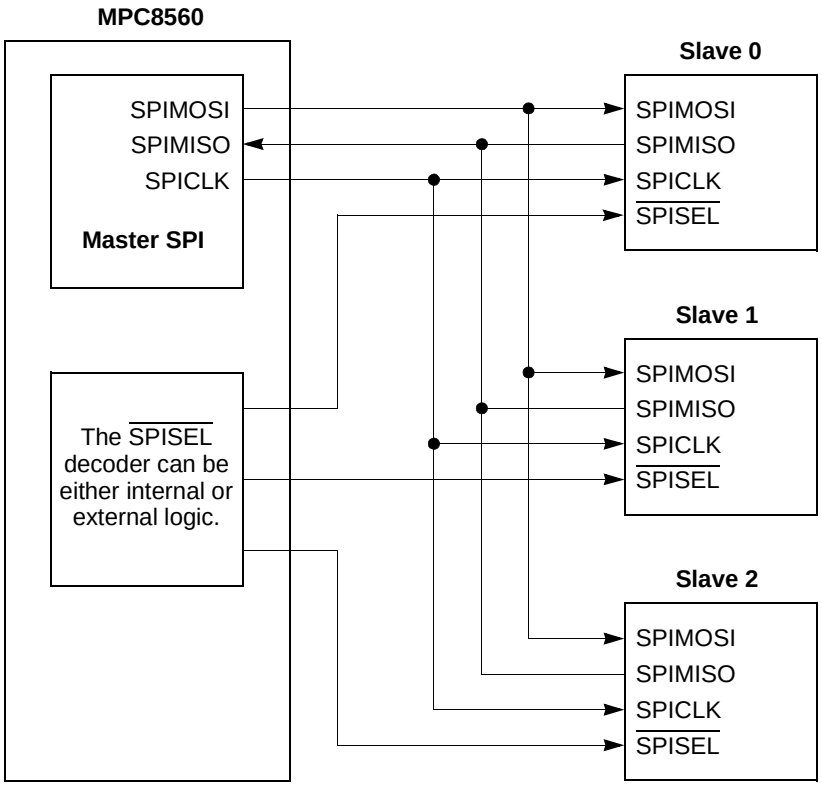


Figure 43-2. Single-Master/Multi-Slave Configuration

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 CPM interrupt controller.

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.

43.3.2 The SPI as a Slave Device

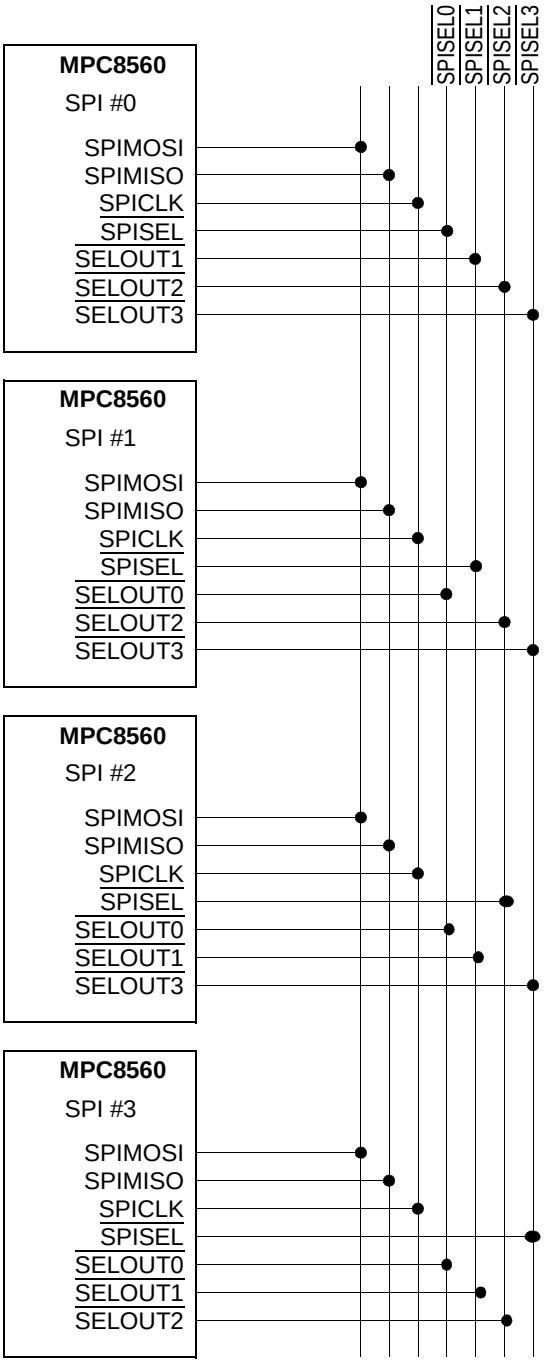
In slave mode, the SPI receives messages from an SPI master and sends a simultaneous reply. The slave's $\overline{\text{SPISEL}}$ must be asserted before Rx clocks are recognized; once $\overline{\text{SPISEL}}$ is asserted, SPICLK becomes an input from the master to the slave. SPICLK can be any frequency 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.

43.3.3 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 43-3. 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 43-3. 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.

43.4 Programming the SPI Registers

The following sections describe the registers used in configuring and operating the SPI.

43.4.1 SPI Mode Register (SPMODE)

The SPI mode register (SPMODE), shown in Figure 43-4, 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 43-4. SPMODE—SPI Mode Register

Table 43-1 describes the SPMODE fields.

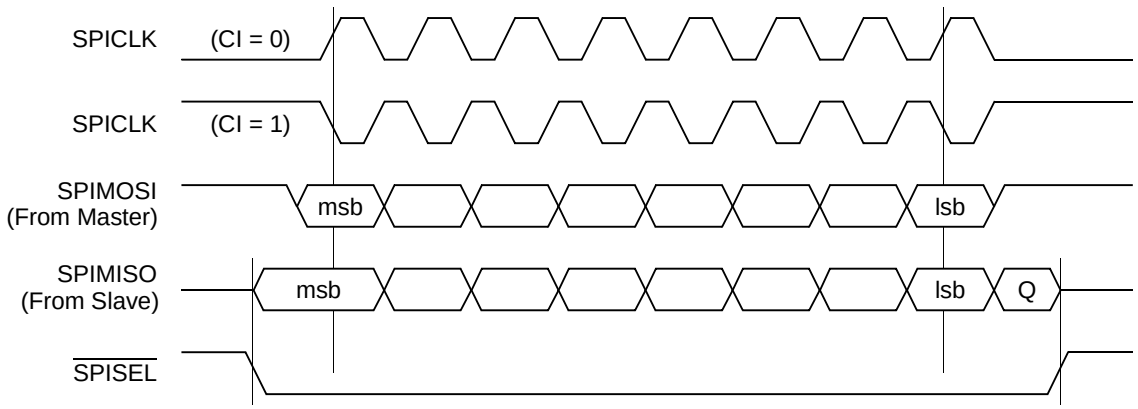
Table 43-1. SPMODE Field Descriptions

Bits	Name[1]	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 43-5 and Figure 43-6. 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 43-5 and Figure 43-6. 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 43-1. SPMODE Field Descriptions (continued)

Bits	Name[1]	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 45.2, “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 43.4.1.1, “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.

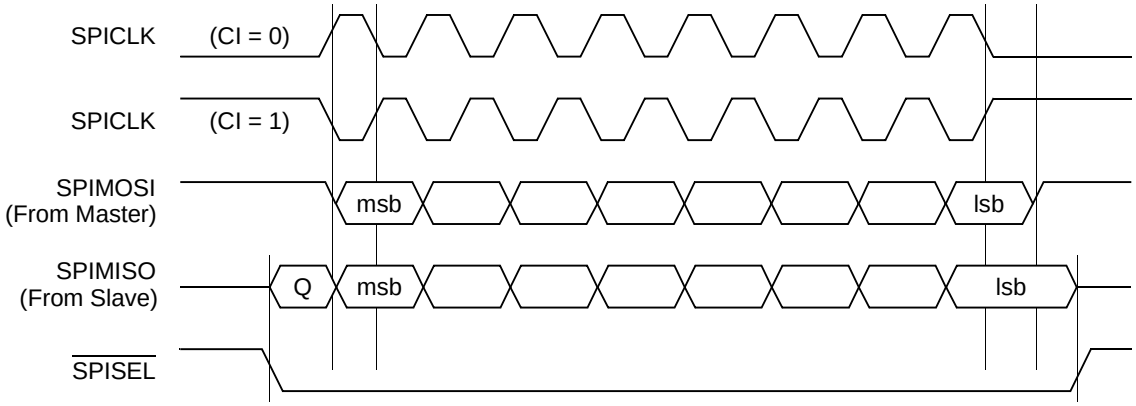
[1] **Boldfaced** entries must be initialized by the user.



NOTE: Q = Undefined Signal.

Figure 43-5. SPI Transfer Format with SPMODE[CP] = 0

Figure 43-6 shows the SPI transfer format in which SPICLK starts toggling at the beginning of the transfer (SPMODE[CP] = 1).



NOTE: Q = Undefined Signal.

Figure 43-6. SPI Transfer Format with SPMODE[CP] = 1

43.4.1.1 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 43-2 are used in the examples.

Table 43-2. 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
```

43.4.2 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 43-7 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 43-7. SPIE/SPIM—SPI Event/Mask Registers

Table 43-3 describes the SPIE/SPIM fields.

Table 43-3. 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.

43.4.3 SPI Command Register (SPCOM)

The SPI command register (SPCOM), shown in Figure 43-8, is used to start SPI operation.

	0	1 … 7
Field	STR	—
Reset	0000_0000	
R/W	Write Only	
Addr	0x9_1AAD	

Figure 43-8. SPCOM—SPI Command Register

Table 43-4 describes the SPCOM fields.

Table 43-4. 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.

43.5 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 43-5 shows the memory map of the SPI parameter RAM.

Table 43-5. SPI Parameter RAM Memory Map

Offset [1]	Name[2]	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 43.5.1, "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.[3] Reserved for CP use.
0x0C	—	Word	The Rx internal data pointer [3] 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 [3] 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.[3] Reserved for CP use.
0x18	TSTATE	Word	Tx internal state.[3] Reserved for CP use.
0x1C	—	Word	The Tx internal data pointer[3] 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[3] is a down-count value initialized with TxBD[Data Length] and decremented with every byte read by the SDMA channels.

Table 43-5. SPI Parameter RAM Memory Map (continued)

Offset [1]	Name[2]	Width	Description
0x24	—	Word	Tx temp.[3] Reserved for CP use.
0x34	—	Word	SDMA temp.

1 From the pointer value programmed in SPI_BASE at CCSRBAR + 0x89FC.
2 **Boldfaced** entries must be initialized by the user.
3 Normally, these parameters need not be accessed. They are listed to help experienced users in debugging.

43.5.1 Receive/Transmit Function Code Registers (RFCR/TFCR)

Figure 43-9 shows the fields in the receive/transmit function code registers (RFCR/TFCR).

	0	1	2	3	4	5	6	7
Field	—		**GBL**	**BO**		**—**	**DTB**	**—**
Reset	0000_0000							
R/W	R/W							
Addr	SPI Base + 04 (RFCR)/SPI Base + 05 (TFCR)							

Figure 43-9. RFCR/TFCR—Function Code Registers

Table 43-6 describes the RFCR/TFCR fields.

Table 43-6. RFCR/TFCR Field Descriptions

Bits	Name[1]	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 Reserved 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.

1 **Boldfaced** entries must be initialized by the user.

43.6 SPI Commands

Table 43-7 lists transmit/receive commands sent to the CP command register (CPCR).

Table 43-7. 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.

43.7 The SPI Buffer Descriptor (BD) Table

As shown in Figure 43-10, BDs are organized into separate RxBD and TxBD tables in dual-port RAM. The tables have the same basic configuration as 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.

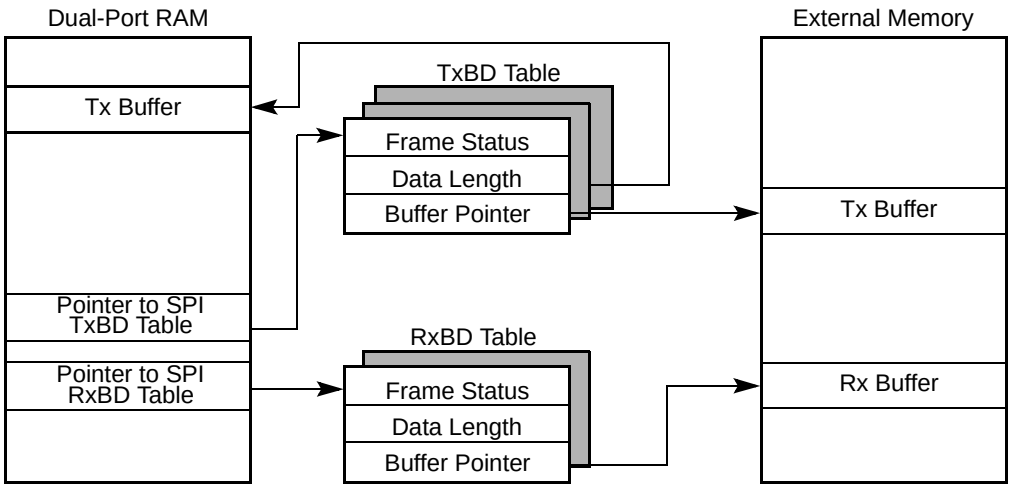


Figure 43-10. SPI Memory Structure

43.7.1 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 43-11 and Figure 43-12, 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, in which case it must be even. The buffer can be in internal or external memory.

43.7.1.1 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 43-11.

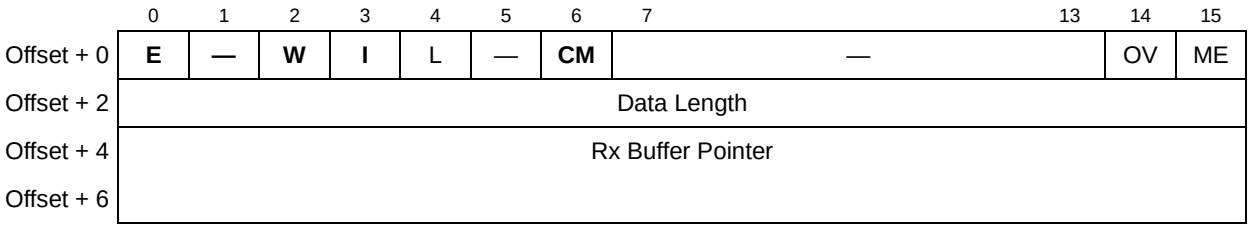


Figure 43-11. SPI RxBD

Table 43-8 describes the RxBD status and control fields.

Table 43-8. SPI RxBD Status and Control Field Descriptions

Bits	Name[1]	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.

[1] **Boldfaced** entries must be initialized by the user.

43.7.1.2 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 a TxBD is shown in Figure 43-12.

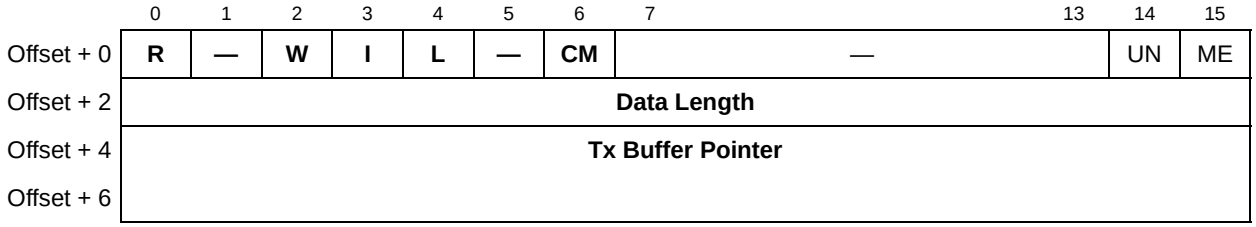


Figure 43-12. SPI TxBD

Table 43-9 describes the TxBD status and control fields.

Table 43-9. SPI TxBD Status and Control Field Descriptions

Bits	Name[1]	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.

[1] **Boldfaced** entries must be initialized by the user.

43.8 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.

43.9 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 43-2).

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.

43.10 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 44
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 44-1, 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.

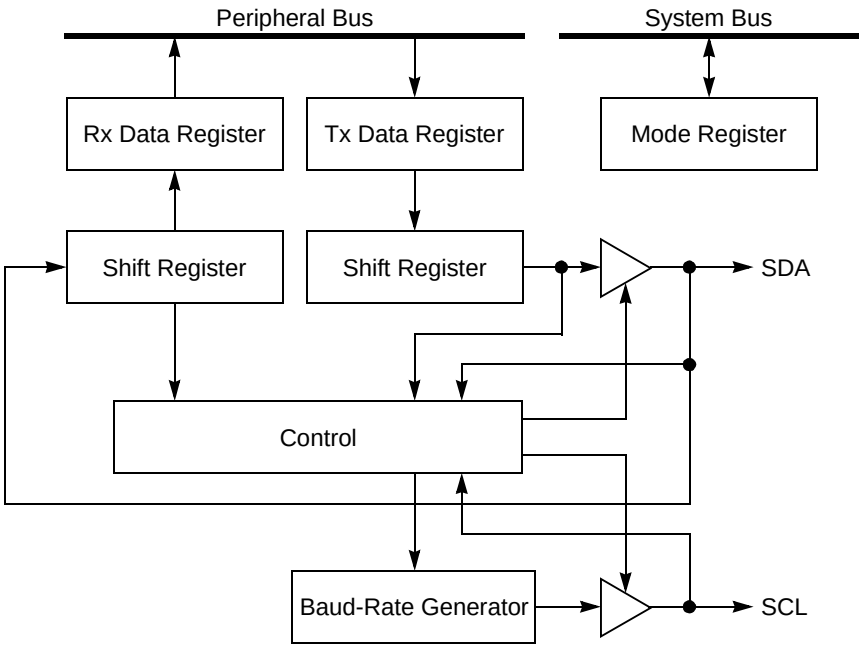


Figure 44-1. I²C Controller Block Diagram

The I²C 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 I²C is not enabled in the I²C mode register (I2MOD[EN] = 0), it consumes little power.

44.1 Features

The following is a list of the I2C controller's main features:

- Two-signal interface (SDA and SCL)
- Support for master and slave I2C operation
- Multiple-master environment support
- Continuous transfer mode for automatic scanning of a peripheral
- Supports a maximum clock rate of 2080 kHz (with a CPM utilization of 25%), assuming a 100-MHz system clock.
- Independent, programmable baud-rate generator
- Supports 7-bit I2C addressing
- Open-drain output signals allow multiple master configuration
- Local loopback capability for testing

44.2 I2C Controller Clocking and Signal Functions

The I2C 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 I2C 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 I2C master/slave configuration is shown in Figure 44-2.

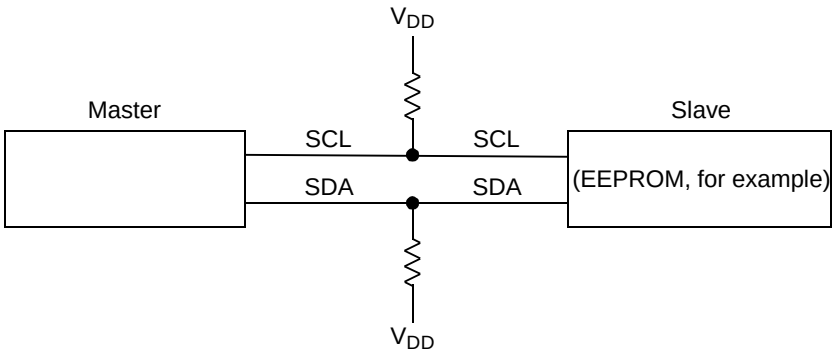


Figure 44-2. I2C Master/Slave General Configuration

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.

44.3 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 44.4.1, "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 44-3.

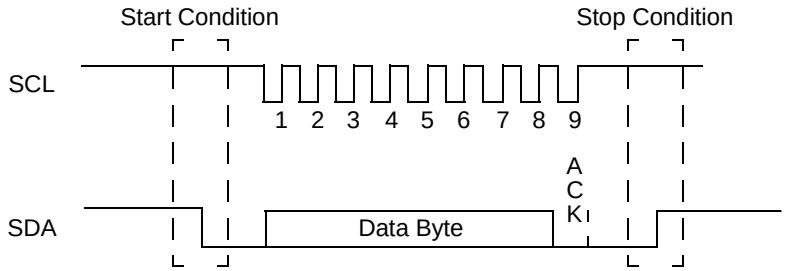


Figure 44-3. I²C Transfer Timing

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.

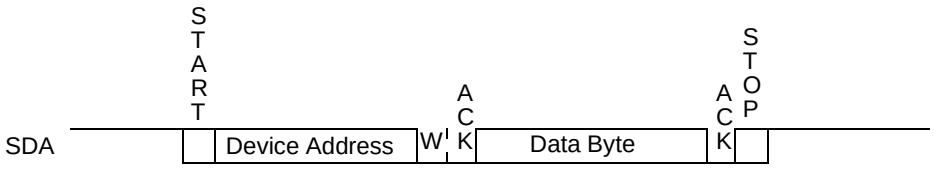
If a master or slave transmitter's current TxBD[L] is set, transmission stops once the buffer is sent; that is, I2COM[STR] must be set again to reactivate transfers. If TxBD[L] is zero, once the current buffer is sent, the controller begins processing the next TxBD without waiting for I2COM[STR] to be set again.

The following sections further detail the transfer process.

44.3.1 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 44-4 shows the timing for a master write.



Note: Data and ACK are repeated n times.

Figure 44-4. 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.

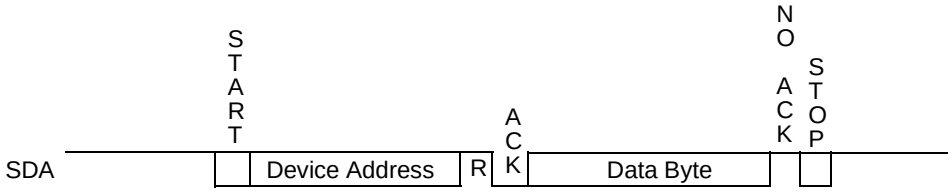
44.3.2 I2C Loopback Testing

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.

44.3.3 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 44-5 shows the timing for a master read.



Note: After the nth data byte, the master does not acknowledge the slave.

Figure 44-5. 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.

44.3.4 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.

44.4 I2C Registers

The following sections describe the I2C registers.

44.4.1 I2C Mode Register (I2MOD)

The I2C mode register, shown in Figure 44-6, controls the I2C modes and clock source.

	0	1	2	3	4	5	6	7
Field	—		REVD	GCD	FLT	PDIV		EN
Reset	0000_0000							
R/W	R/W							
Addr	0x9_1860							

Figure 44-6. I2C Mode Register (I2MOD)

Table 44-1 describes I2MOD bit functions.

Table 44-1. 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 44-1. 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. The I²C is in a reset state and consumes minimal power. 1 I²C is enabled. Do not change other I2MOD bits when EN is set.

44.4.2 I²C Address Register (I2ADD)

The I²C address register, shown in Figure 44-7, holds the address for this I²C port.

	0	6	7
Field	SAD		—
Reset	Undefined		
R/W	R/W		
Addr	0x9_1864		

Figure 44-7. I²C Address Register (I2ADD)

Table 44-2 describes I2CADD fields.

Table 44-2. 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.

44.4.3 I2C Baud Rate Generator Register (I2BRG)

The I2C baud rate generator register, shown in Figure 44-8, sets the divide ratio of the I2C BRG.

	0	7
Field	DIV	
Reset	1111_1111	
R/W	R/W	
Addr	0x9_1868	

Figure 44-8. I2C Baud Rate Generator Register (I2BRG)

Table 44-3 describes I2BRG fields.

Table 44-3. 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.

44.4.4 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 44-9 shows both registers.

	0	2	3	4	5	6	7
Field	—		TXE	—	BSY	TXB	RXB
Reset	0000_0000						
R/W	R/W						
Addr	0x9_1870(I2CER)/0x9_1874 (I2CMR)						

Figure 44-9. I2C Event/Mask Registers (I2CER/I2CMR)

Table 44-4 describes the I2CER/I2CMR fields.

Table 44-4. 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.

Table 44-4. I2CER/I2CMR Field Descriptions

Bits	Name	Description
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.

44.4.5 I2C Command Register (I2COM)

The I2C command register, shown in Figure 44-10, 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 44-10. I2C Command Register (I2COM)

Table 44-5 describes I2COM fields.

Table 44-5. 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.

44.5 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 44-6 describes the I2C parameter RAM memory map.

Table 44-6. I2C Parameter RAM Memory Map

Offset [1]	Name[2]	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 44-11 and Table 44-7.
0x05	**TFCR**	Byte	
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.[3] Reserved for CP use.
0x0C	RPTR	Word	Rx internal data pointer [3] 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 [3] 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. [3] Reserved for CP use.
0x18	TSTATE	Word	Tx internal state. [3] Reserved for CP use.
0x1C	TPTR	Word	Tx internal data pointer [3] 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.

Table 44-6. I²C Parameter RAM Memory Map (continued)

Offset [1]	Name[2]	Width	Description
0x22	TCOUNT	Hword	Tx internal byte count [3] 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. [3] Reserved for CP use.
0x34	SDMATMP	Word	SDMA temp. [3] Reserved for CP use.

1 From the pointer value programmed in I2C_BASE at CCSRBAR + 0x8AFC.
2 **Boldfaced** entries must be initialized by the user.
3 Normally, these parameters need not be accessed.

Figure 44-11 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 44-11. I²C Function Code Registers (RFCR/TFCR)

Table 44-7 describes the RFCR/TFCR bit fields.

Table 44-7. RFCR/TFCR Field Descriptions

Bits	Name[1]	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.

1 **Boldfaced** entries must be initialized by the user.

44.6 I2C Commands

The I2C transmit and receive commands, shown in Table 44-8, are issued to the CP command register (CPCR).

Table 44-8. 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.

44.7 I2C Buffer Descriptor (BD) Tables

As shown in Figure 44-12, 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.

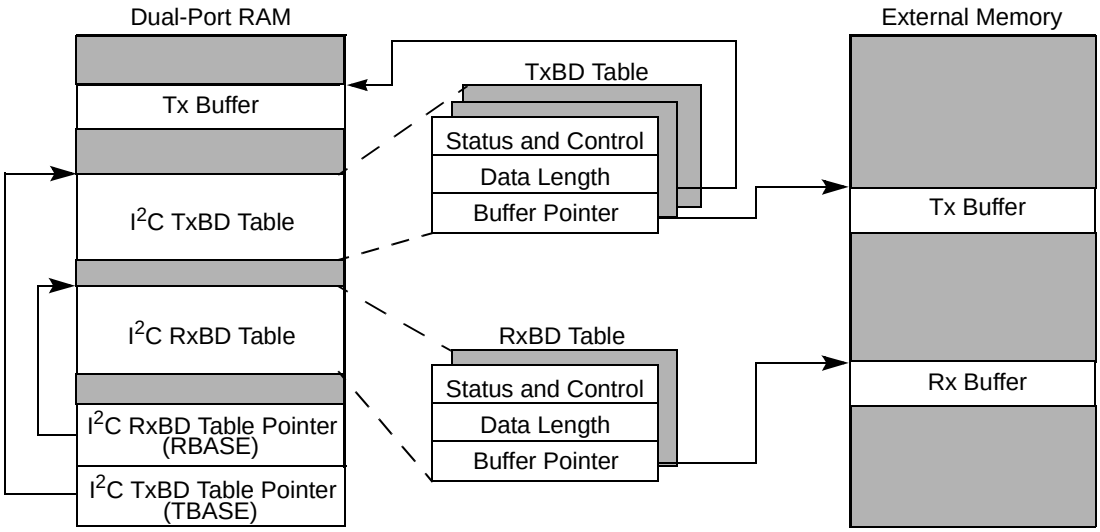


Figure 44-12. I2C Memory Structure

44.8 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 44-13 and Figure 44-14, 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.

44.8.1 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.

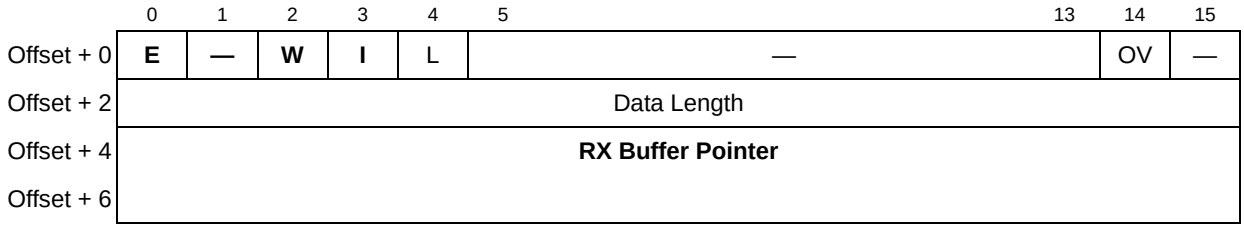


Figure 44-13. I2C RxBD

Table 44-9 describes I2C RxBD status and control bits.

Table 44-9. I2C RxBD Status and Control Bits

Bits	Name[1]	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.

1 **Boldfaced** entries must be initialized by the user.

44.8.1.1 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 44-14, contains status and control bits.

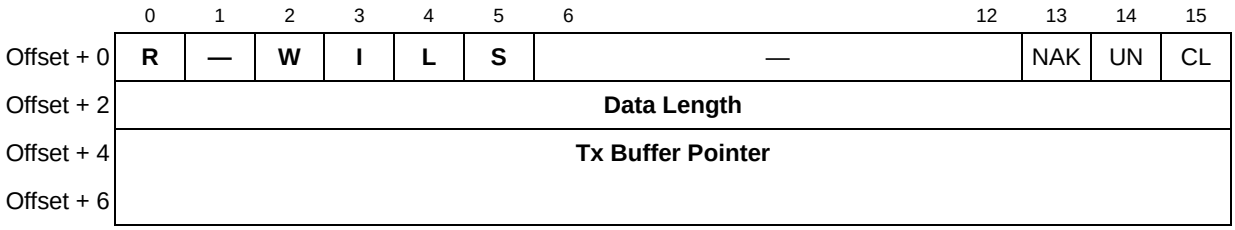


Figure 44-14. I2C TxBD

Table 44-10 describes I2C TxBD status and control bits.

Table 44-10. I2C TxBD Status and Control Bits

Bits	Name[1]	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.

1 **Boldfaced** entries must be initialized by the user.

Chapter 45
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.

45.1 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

45.2 Port Registers

Each port has four memory-mapped, read/write, 32-bit control registers.

45.2.1 Port Open-Drain Registers (PODRA–PODRD)

The port open-drain register (PODR), shown in Figure 45-1, 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 45-1. Port Open-Drain Registers (PODRA–PODRD)

Table 45-1 describes PODR fields.

Table 45-1. 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.

45.2.2 Port Data Registers (PDATA–PDATD)

A read of a port data register (PDAT*x*), shown in Figure 45-2, 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 45-2. Port Data Registers (PDATA–PDATD)

45.2.3 Port Data Direction Registers (PDIRA–PDIRD)

The port data direction register (PDIR), shown in Figure 45-3, 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 45-3. Port Data Direction Register (PDIR)

Table 45-2 describes PDIR fields.

Table 45-2. 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.

45.2.4 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 45-4. Port Pin Assignment Register (PPARA–PPARD)

Table 45-3 describes PPAR*x* fields.

Table 45-3. 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.

45.2.5 Port Special Options Registers A–D (PSORA–PSORD)

Figure 45-5 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 45-5. Special Options Registers (PSORA–POSRD)

PSOR bits are effective only if the corresponding PPAR*x*[DD*x*] = 1 (a dedicated peripheral function). Table 45-4 describes PSOR*x* fields.

Table 45-4. PSOR*x* Field Descriptions

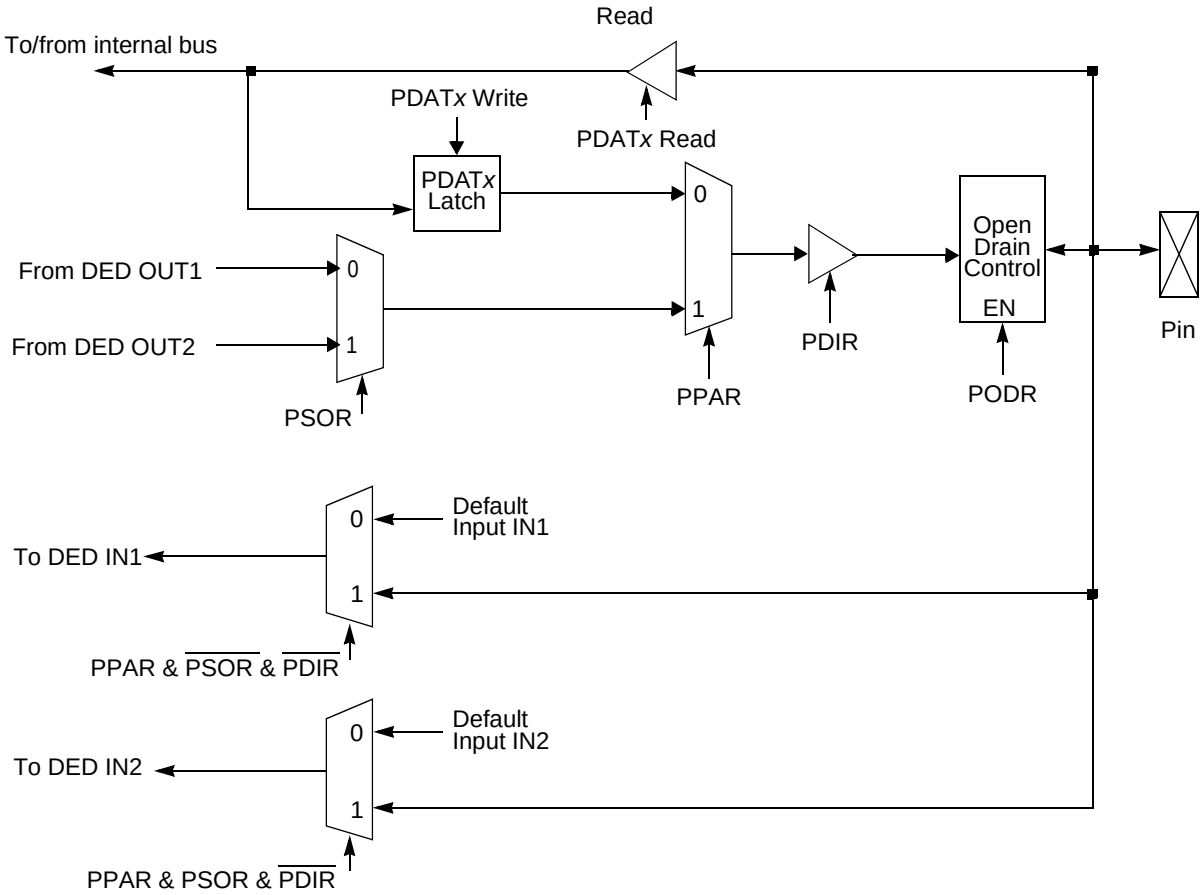
Bits	Name	Description
0–31	SO*x*	Special-option. Determines whether a pin configured for a dedicated function (PPAR*x*[DD*x*] = 1) uses option 1 or option 2. Note that bits SO0–SO3 are valid for PSORA and PSORC only. Options are described in Section 45.2, “Port Registers.” 0 Dedicated peripheral function. 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.

45.3 Port Block Diagram

Figure 45-6 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 45-6. Port Functional Operation

45.4 Port Pins Functions

Each pin can operate as a general purpose I/O pin or as a dedicated input or output pin.

45.4.1 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.

45.4.2 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 45.5, “Ports Tables.”

45.5 Ports Tables

Table 45-5 through Table 45-8 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 45-7, 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.

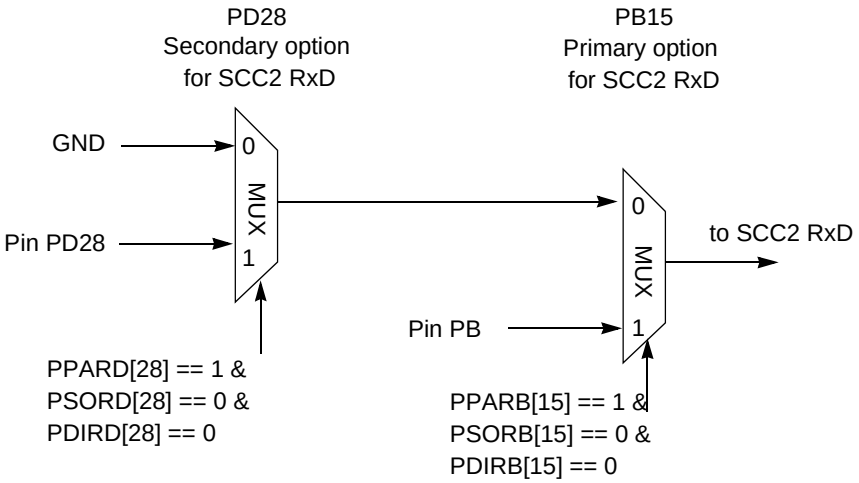


Figure 45-7. Primary and Secondary Option Programming

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 45-5 shows the port A pin assignments.

Table 45-5. 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 I/O 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 45-5. 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 I/O 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 45-5. 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 I/O 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 45-5. 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 I/O 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** I/O, 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[0–5] 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 26.1.2, “SDMA Event Registers (SMEVR and LMEVR).”

2 Available only when the primary option for this function is not used.

Table 45-6 shows the port B pin assignments.

Table 45-6. 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 I/O if Specified)	Default Input
PB31	**FCC2: TX_ER** MII	**FCC2: RxSOC** UTOPIA	GND		**TDM_B2: L1TXD** I/O	GND
PB30	**FCC2: TxSOC** UTOPIA	**FCC2: RX_DV** MII **FCC2: CRS_DV** RMII	GND		**TDM_B2: L1RXD** I/O	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** I/O	GND
PB26	**FCC2: TxD[1]** UTOPIA 8	**FCC2: CRS** MII	GND		**TDM_C2: L1RXD** I/O	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** I/O	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** I/O	GND

Table 45-6. 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 I/O 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** I/O (primary option)	by PD28
PB14	**FCC3: TX_EN** MII/RMII	**SCC3: RXD** (primary option)	by PD25		**TDM_C1: L1RXD** I/O (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

Table 45-6. 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 I/O if Specified)	Default Input
PB11	**FCC2: TxD[0]** UTOPIA 8	**FCC3: RxD[3]** MII/HDLC nibble	GND		**TDM_D1: L1TXD** I/O (primary option)	by PD25
PB10	**FCC2: TxD[1]** UTOPIA 8	**FCC3: RxD[2]** MII/HDLC nibble	GND		**TDM_D1: L1RXD** I/O (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** I/O, 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** I/O, 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 45-7 shows the port C pin assignments.

Table 45-7. 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 I/O 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] I/O (secondary option)	
PC18		CLK14	GND		**timer3/4: $\overline{\text{TGATE2}}$**	GND
PC17	**BRG8: BRGO**	CLK15/TIN3	GND			
PC16		CLK16/TIN4	GND			

Table 45-7. 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 I/O if Specified)	Default Input
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
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

Table 45-7. 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 I/O if Specified)	Default Input
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
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: $\overline{\text{SPISEL}}$**[2] I/O (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 and 4 (master mode) can come from FCC2, depending on CMXUAR programming. (See Section 23.4.1, “CMX UTOPIA Address Register (CMXUAR).”)

Table 45-8 shows the port D pin assignments.

Table 45-8. 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 I/O if Specified)	Default Input
PD31		**SCC1: RXD**	GND			
PD30	**FCC2: TxEnb** UTOPIA master	**FCC2: TxEnb** UTOPIA slave	GND	**SCC1: TXD**		

Table 45-8. 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 I/O if Specified)	Default Input
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] I/O (secondary option)	GND
PD27	**SCC2: TXD**	**FCC1: RxD[7]** UTOPIA 16	GND		**TDM_C1: L1RXD**[3] I/O (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] I/O (secondary option)	GND
PD24	**SCC3: TXD**	**FCC1: RxD[5]** UTOPIA 16	GND		**TDM_D1: L1RXD**[3] I/O (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] I/O, serial (secondary option)	GND
PD21	**SCC4: TXD**	**FCC1: RxD[3]** UTOPIA 16	GND		**TDM_A2: L1RXD**[3] I/O, 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

Table 45-8. 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 I/O if Specified)	Default Input
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: 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** I/O (primary option)	GND
PD17	**BRG2: BRGO**	**FCC1: RxPrty** UTOPIA (primary option)	GND		**SPI: SPIMOSI** I/O	V_{DD}
PD16	**FCC1: TxPrty** UTOPIA (primary option)	**TDM_C1: L1TSYNC/GRANT**[3] (secondary option)	GND		**SPI: SPIMISO** I/O	SPIMOSI
PD15	**TDM_C2: $\overline{\text{L1RQ}}$**	**FCC1: RxD[1]** UTOPIA 16	GND		**I2C: I2CSDA** I/O	V_{DD}
PD14	**TDM_C2: L1CLKO**	**FCC1: RxD[0]** UTOPIA 16	GND		**I2C: I2CSCL** I/O	GND
PD13	**SI1: L1ST1**				**TDM_B1: L1TXD** I/O	GND
PD12	**SI1: L1ST2**				**TDM_B1: L1RXD** I/O	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

Table 45-8. 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 I/O if Specified)	Default Input
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.

45.6 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.

Appendix A
Revision History

This appendix provides a list of the major differences between revisions of the *MPC8560 PowerQUICC III Integrated Communications Processor Reference Manual, Revision 0* and *MPC8560 PowerQUICC III Integrated Communications Processor Reference Manual, Revision 1*.

A.1 Changes From Revision 0 to Revision 1

Major changes to the *MPC8560 PowerQUICC III Integrated Communications Processor Reference Manual* between Revision 0 to Revision 1 are as follows:

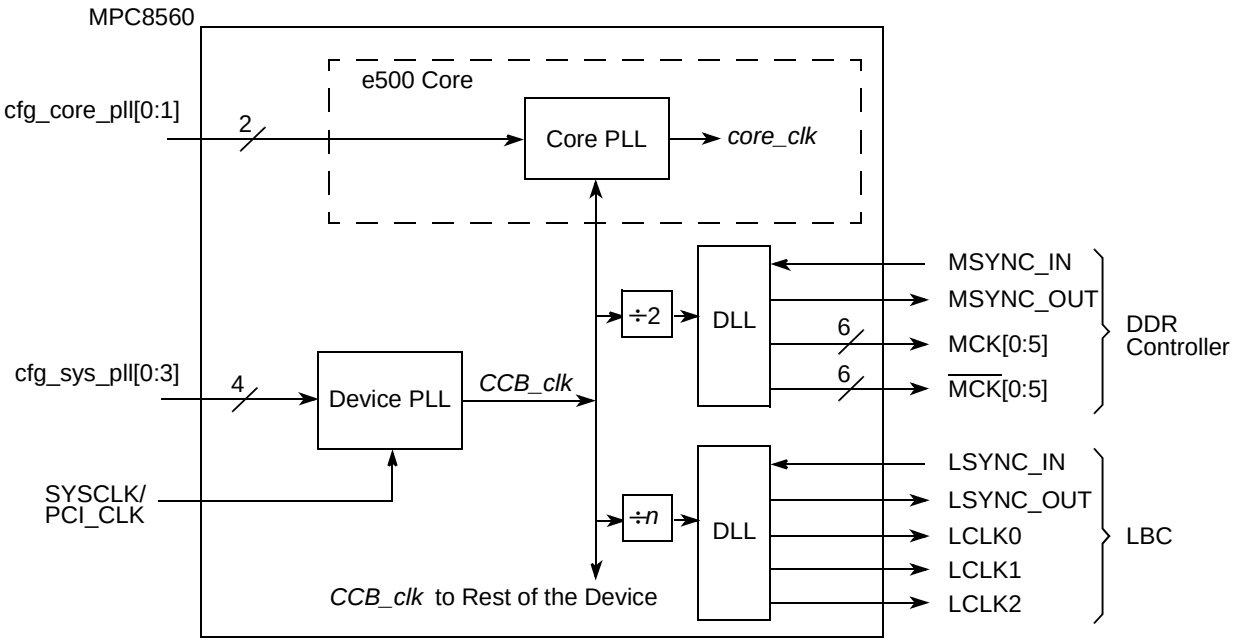
Section, Page	Changes
Chapters 4, 15, 17, 19	Throughout all applicable chapters, correctly state the debug mode source ID (formerly documented to be visible on signals PCI_AD[63:59]) to be visible on signals PCI_AD[62:58].
2.4, 2-19	In Table 2-9, correct the reset value for the I2CDFSRR register to be 0x10.
2.4, 2-28	In Table 2-9, add the following rows under the TSEC1 FIFO Control and Status Registers section:

0x2_404C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	13.5.3.2.1/13-30
0x2_4050–0x2_4088	Reserved	R	0x0000_0000	—

2.4, 2-28	In Table 2-9, correct the reset value for the OSTBD register to be 0x0800_0000.
2.4, 2-29	In Table 2-9, correct the MACCFG1 register from having R/W, R access to R/W access.
2.4, 2-29	In Table 2-9, correct the reset value for the IFSTAT register to be 0x0000_0000.
2.4, 2-31	In Table 2-9, correct the CAR1 and CAR2 registers from having R access to R/W access. Also, remove the “Cleared on read” footnote designation from the CAR1 and CAR2 registers.

2.4, 2-32	In Table 2-9, correct the offset for the Reserved addresses after the EOI register to be 0x4_00C0–0x4_0FF0.
2.4, 2-59	In Table 2-9, correct the reset value for the GPINDR register to be 0x*nnnn*_0000.
2.4, 2-60	In Table 2-9, correct the name of the register with offset 0xE_1094 from PMLCB5 to PMLCB8.
2.4, 2-54–58	In Table 2-9, correct the access designation for all RapidIO Registers listed as “Special” to read “R/W”.
3.1, 3-3	In Figure 3-1, correct the TEST_SEL test signal to be active low, “$\overline{\text{TEST\_SEL}}$.”
3.1, 3-9	In Table 3-1, correct the TEST_SEL signal to be active low, “$\overline{\text{TEST\_SEL}}$”.
3.1, 3-13	In Table 3-2, correct the TEST_SEL signal to be active low, “$\overline{\text{TEST\_SEL}}$”.
4.3.1.3, 4-7	Replace the second sentence of the first paragraph to read: “The first instruction executed by the e500 core is always address 0xFFFF_FFFC, which must be a branch to an address within the 4-Kbyte boot page.”
4.4.3.3, 4-13	Insert the following sentence at the start of the first paragraph: “The first instruction executed by the e500 core is always address 0xFFFF_FFFC, which must be a branch to an address within the 4-Kbyte boot page.”
4.4.3.3, 4-14	Add the following sentence to the end of the last paragraph: “If enabled, this translation only affects CPU accesses to 0xFFFF_F*nnn*.”
4.4.4.1, 4-22	Append the end of the first paragraph of Section 4.4.4.1, with the following sentence: “Note that the divide-by-two CCB clock divider and the divide-by-*n* CCB clock divider, shown in Figure 4-6, are located in the DDR and local bus blocks, respectively.”

4.4.4.1, 4-23 Replace the existing Figure 4-6 with the following figure (Note the addition of the CCB clock dividers prior to the DDR DLL and the LBC DLL):



4.4.4.4, 4-25 In Figure 4-8, replace the sentence:

"Watchdog timer events based on one of the 64 TB bits selected by TCR[WP] concatenated with the EIS-defined TCR[WPEXT] (WP||WPEXT)."

with the following sentence:

"Watchdog timer events based on one of the 64 TB bits selected by concatenating TCR[WPEXT] with the EIS-defined TCR[WP] (WPEXT||WP)."

4.4.4.4, 4-25 In Figure 4-8, replace the sentence:

"Fixed-interval timer events based on one of the 64 TB bits selected by TCR[FP] concatenated with the EIS-defined TCR[FPEXT] (FP||FPEXT)."

with the following sentence:

"Fixed-interval timer events based on one of the 64 TB bits selected by concatenating TCR[FPEXT] with the EIS-defined TCR[FP] (FPEXT||FP)."

5.5, 5-13	Prior to Section 5.5.1, insert a new section, “Initial Instruction Fetch,” consisting of the following text (former Section 5.5.1 now becomes Section 5.5.2.): “The e500 core begins execution at fixed virtual address 0xFFFF_FFFC. The MMU has a default page translation which maps this to the identical physical address. So, the instruction at physical address 0xFFFF_FFFC must be a branch to another address within the 4-Kbyte boot page.”
5.6, 5-17	Remove the registers IAC3 and IAC4 from Figure 5-6.
5.14, 5-32	In Table 5-8, replace the HID1 implementation description of HID1[RFXE] with the following:

HID1[RFXE] controls whether assertion of *core_fault_in* causes a machine check interrupt. Assertion of *core_fault_in* can result from uncorrectable data error, such as an L2 multibit ECC error. It can also occur for a system error if logic on the integrated device signals a fault for nonfatal errors (read data is corrupt (or zero), but the transaction can complete without corrupting other system state, making it unnecessary to take a machine check (for example, a master abort of a PCI transaction).
If RFXE is 0, and *core_fault_in* is asserted, any data on the CCB is dropped, either stalling the load/store unit and causing the e500 pipeline to stall until an interrupt occurs (typically generated by the programmable interrupt controller (PIC) in response to the fault) or allowing processingþto continue with the bad data until the interrupt occurs. Because *core_fault_in* cannot cause a machine check if RFXE is 0, it is critical that the system be configured to generate the appropriate interrupt, as described below.
It is also possible to hang the processor (requiring a hard reset to recover) if a guardedþload hits in the L2 cache and gets an uncorrectable ECC error. Because of this, avoid defining memory as cacheable but guarded. If this combination is required, RFXE must be enabled, in which case an error causes both a machine check interrupt and an external interrupt when a bus fault condition is detected unless interrupts are masked for all sources of bus faults.
If RFXE is 0, conditions that cause the assertion of core_fault_in cannot directly cause the e500 to generate a machine check; however, PowerQUICC III devices must be configured to detect and enable such conditions. The following describes how error bits should be configured:

- ECM mapping errors: EEER[LAEE] must be set. See Section 8.2.1.4, “ECM Error Enable Register (EEER).”
- L2 multiple-bit ECC errors: L2ERRDIS[MBECCDIS] must be cleared to ensure that error can be detected. L2ERRINTEN[MBECCINTEN] must be set. See Section 7.3.1.5, “L2 Error Registers.”
- DDR multiple-bit ECC errors. ERR_DISABLE[MBED] and ERR_INT_EN[MBEE] must be zero and DDR_SDRAM_CFG[ECC_EN] must be one to ensure that an interrupt is generated. See Section 9.4.1, “Register Descriptions.”
- PCI. The appropriate parity detect and master-abort bits in ERR_DR must be cleared and the corresponding enable bits in ERR_EN must be set to ensure that an interrupt is generated. Section 15.3.1.4, “PCI/X Error Management Registers.”
- Local bus controller parity errors. LTEDR[PARD] must be cleared and LTEIR[PARI] must be set to ensure that an parity errors can generate an interrupt. See Section 12.3.1.11, “Transfer Error Check Disable Register (LTEDR),” and Section 12.3.1.12, “Transfer Error Interrupt Enable Register (LTEIR).”
-
- RapidIO. PCR[CCE] must be set to ensure that an interrupt is generated due to a CRC error. See Section 16.3.2.1.2, “Port Configuration Register (PCR).”

RFXE must also be set if software requires that code execution stop immediately when a bus fault occurs rather than continuing with the bad data until the interrupt arrives. Again, this results in both a machine check interrupt and an external interrupt when a bus fault is detected, unless all possible sources for bus fault have their interrupts masked.þThe machine check interrupt can then reenable normal interrupts and wait for the interrupt due to the fault to be received before returning from the machine check.

6.1.1, 6-2	Remove IAC3 and IAC4 from Figure 6-1 as they are not supported.

6.6.1, 6-14 — In Table 6-8, change the second sentence of the description for bits 32–33 of the TCR register to the following:

WPEXT[0–3] || WP[0–1] = 0b00_0000 selects TBU[32] (the msb of the TB)

WPEXT[0–3] || WP[0–1] = 0b11_1111 selects TBL[63] (the lsb of the TB)

6.6.1, 6-14 — In Table 6-8, change the description of TCR[WRC] as follows:

01 A second timeout is ignored, regardless of the value of MSR[ME].

6.6.1, 6-14 — In Table 6-8, change the second sentence of the description for bits 38–39 of the TCR register to the following:

FPEXT[0–3] || FP[0–1] = 0b00_0000 selects TBU[32] (the msb of the TB)

FPEXT[0–3] || FP[0–1] = 0b11_1111 selects TBL[63] (the lsb of the TB)

6.7.2.4, 6-22 — In Table 6-12, change the bit range “43–46” to “36–46”.

6.10.2, 6-26 — In Table 6-17, replace the description of HID1[RFXE] with the following:

Read fault exception enable. Controls whether assertion of *core_fault_in* causes a machine check interrupt. The assertion of *core_fault_in* can result from an L2 multibit ECC error. It can also occur for a system error if logic on the integrated device signals a fault for nonfatal errors (read data is corrupt (or zero), but the bus transaction can complete without corrupting other system state, making it unnecessary to take a machine check (for example, a master abort of a PCI transaction).

0 Assertion of *core_fault_in* cannot cause a machine check. RFXE should be left clear if an interrupt is to be reported by the integrated device through *int* or *c_int* for this condition. If RFXE = 0, it is important that the integrated device generates an interrupt if *core_fault_in* is asserted.
If *core_fault_in* is asserted, any data on the CCB is dropped, stalling the load/store unit and eventually causing the e500 pipeline either to stall until an interrupt occurs (typically generated by the programmable interrupt controller (PIC) in response to the fault) or to continue processing with bad data untill the interrupt occurs. Because *core_fault_in* cannot cause a machine check, if RFXE is 0, it is critical that the system be configured to generate the appropriate interrupt.
It is also possible to hang the processor (requiring a hard reset to recover) if a guarded load hits in the L2 cache and gets an uncorrectable ECC error. Because of this, avoid defining memory as cacheable but guarded. If this combination is required, RFXE must be enabled, in which case an error causes both a machine check interrupt and an external interrupt when a bus fault condition is detected unless interrupts are masked for all sources of bus faults, such as DRAM ECC errors, PCI parity errors, local bus parity errors, and others.
RFXE must also be set if software requires that code execution stop immediately when a bus fault occurs rather than continuing with the bad data until the interrupt arrives. Again, this results in both a machine check interrupt and an external interrupt when a bus fault is detected, unless all possible sources for bus fault have their interrupts masked.þThe machine check interrupt can then reenable normal interrupts and wait for the interrupt due to the fault to be received before returning from the machine check.

1 A machine check can occur due to assertion of *core_fault_in*.
If MSR[ME] = 1 and a fault is signaled, a machine check interrupt occurs.
If MSR[ME] = 0 and a fault is signaled, a checkstop occurs.
Note that if RFXE is set and another mechanism is configured to generate an interrupt in response to assertion of *core_fault_in*, the same event causes two interrupts, the machine check enabled by setting RFXE and the interrupt triggered by the on-chip peripheral or other block; therefore, RFXE should be set only if no other mechanism is configured to generate an interrupt for this case.

Note that the L2 cache detects any assertion of *core_fault_in* and ensures that the L2 cache is not corrupted when data is dropped for this type of transaction.

6.11.3, 6-29	In Table 6-20, describe bits 34–38 as ‘Reserved’ as is properly expressed in Figure 6-35. Disregard descriptions associated with bits 34, 35, or 36 in Table 6-20.
6.12.2, 6-32	In Figure 6-38 and Table 6-22, describe bits 59 and 60 as “Reserved”.
6.12.5.3, 6-36	In Table 6-28, describe bits 52–56 as “Reserved for implementation-specific use”.
6.13.3, 6-44	Remove all references to IAC3 and IAC4 from this section.
7.3.1.1, 7-8	In Table 7-2, change the description for bit 1 from “Accesses to memory-mapped SRAM are unaffected...” to “Data in memory-mapped SRAM regions is unaffected...”
7.3.1.1, 7-8 and 7-9	In Figure 7-7 and Table 7-2, show bit 11 to be reserved.
7.3.1.5.2, 7-16 and 7-17	In Table 7-12, change the bit descriptions for bits 27 and 31 of the L2ERRDET register as follows: Bit 27, TPARERR: Add the following text: “Note that if an L2 cache tag parity error occurs on an attempt to write a new line, the L2 cache must be flash invalidated. L2 functionality is not guaranteed if flash invalidation is not performed after a tag parity error.” Bit 31, L2CFGERR: Add the following text to the 1st line of description, “Reports inconsistencies between the L2SIZ, L2BLKSZ, and L2SRAM settings of the L2 control register (L2CTL).”
7.7.4, 7-25	Add the following note after the second paragraph:

NOTE

There is a scenario in which a lock clear operation appears to fail to clear a lock in the L2 cache. This occurs only when the attempt to set the lock results in a bus error (for example, PCI returns an error condition).

Assume the following scenario:

1. The e500 attempts to set a lock in the L2 cache (by executing a **dcbtls** or **icbtls** instruction with CT = 1). The line is not already present in the cache, so it must be read from external memory. This read encounters an error which, depending on the chip configuration, will be reported to the core (probably as an interrupt).

2. At (or near) the same time, a cache external write to the same cache line is being mastered by the ECM.

3. Very soon after the cache external write, a transaction to clear the lock occurs. This can be caused by the processor executing a **dcblc** or **icblc** instruction with CT = 1, or by the ECM mastering a lock clear transaction.

If this scenario occurs within a tight timing window, the cache line may unexpectedly remain locked at the end of the sequence.

The interrupt handler may want to clear the erroneously remaining lock in this case.

7.9, 7-29

Add new Section 7.9.2, “Flash Invalidation of L2 Cache” as follows:
“The L2 cache may be completely invalidated by setting the L2I bit of the L2 control register (L2CTL). Note that no data is lost in this process because the L2 cache is a write-through cache and contains no modified data. Flash invalidation of the cache is necessary when the cache is initially enabled and may be necessary to recover from some error conditions such as a tag parity error.

The invalidation process requires several cycles to complete. The L2I bit remains set during this procedure and is then cleared automatically when the procedure is complete. The L2 cache controller issues retries for all transactions on the e500 core complex bus (CCB) while the flash invalidation process is in progress.

Note that the contents of memory-mapped SRAM regions of the data array are unaffected by a flash invalidation of the L2 cache regions fo the array.

7.10, 7-33

Add new Section 7.10, “Initialization/Application Information,” with the following sections: Section 7.10.1, “Initialization,” with new subsections Section 7.9.1.1, “L2 Cache Initialization,” and Section 7.9.1.2, “Memory-Mapped SRAM Initialization.”

7.9.1.1 L2 Cache Initialization

After power-on reset, the valid bits in the L2 cache status array are in random states. Therefore, it is necessary to perform a flash invalidate operation before using the array as an L2 cache. This is done by writing a 1 to the L2I bit of the L2 control register (L2CTL). This can be done before or simultaneously with the write that enables the L2 cache. That is, the L2E and L2I bits of L2CTL can be set simultaneously. The L2I bit clears automatically, so no further writes are necessary.

7.9.1.2 Memory-Mapped SRAM Initialization

After power-on reset, the contents of the data and ECC arrays and are random, so all SRAM data must be initialized before it is read. If the cache is initialized by the core or any other device that uses sub-cacheline transactions, ECC error checking should be disabled during the initialization process to avoid false ECC errors generated during the read-modify-write process used for sub-cacheline writes to the SRAM array. This is done by setting the multi- and single-bit ECC error disable bits of the L2 error disable register (L2ERRDIS[MBECCDIS, SBECCDIS]). See Section 7.3.1.9.2, “Error Control and Capture Registers.” If the array is initialized by a DMA engine using cache-line writes, then ECC checking can remain enabled during the initialization process.

Also, add new Section 7.10.2, “Managing Errors,” with subsections Section 7.10.2.1, “ECC Errors,” and Section 7.10.2.2, “Tag Parity Errors.”

7.10.2.1 ECC Errors

An individual soft error that causes a single- or mulit-bit ECC error can be cleared from the L2 array simply by executing a **dcbf** instruction for the address captured in the L2ERRADDR register. This will invalidate the line in the L2 cache. When the load that caused the ECC error is performed again, the data will be re-allocated into the L2 with ECC bits set properly again.

If the threshold for single bit errors set in the L2ERRCTL register is exceeded, then the L2 cache should be flash invalidated to clear out all single-bit errors.

Note that no data is lost by executing **dcbf** instructions or flash invalidate operations because the L2 cache is write-through and contains no modified data.

7.10.2.2 Tag Parity Errors

A tag parity error must be fixed by flash invalidating the L2 cache. Note that executing a **dcbf** instruction for the address that caused the error to be reported is not sufficient because a tag parity error is seen as an L2 miss and does not cause invalidation of the bad tag. Proper L2 operation cannot be

guaranteed if an L2 tag parity error is not repaired by a flash invalidation of the entire array.

9.3.2.1, 9-6 — In Table 9-3, change the Timing description for the MA[0:14] signals to read:

"Assertion/Negation—The address is always driven when the memory controller is active. It is valid when a transaction is driven to DRAM (when $\overline{\text{MCS}n}$ is negated).

High-impedance—When the memory controller is idle."

9.3.2.1, 9-6 — In Table 9-3, change the Timing description for the $\overline{\text{MCAS}}$ signal to read:

"Assertion/Negation—Assertion and negation timing is directed by the values described in Section 9.4.1.3, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)."

High-impedance—$\overline{\text{MCAS}}$ is always driven unless the memory controller is idle."

9.3.2.1, 9-7 — In Table 9-3, change the Timing description for the $\overline{\text{MRAS}}$ signal to read:

"Assertion/Negation—Timing is directed by the values described in Section 9.4.1.3, "DDR SDRAM Timing Configuration 1 (TIMING_CFG_1)."

High-impedance—$\overline{\text{MRAS}}$ is always driven unless the memory controller is idle."

9.3.2.1, 9-7 — In Table 9-3, change the Timing description for the $\overline{\text{MWE}}$ signal to read:

"Assertion/Negation—Similar timing as $\overline{\text{MRAS}}$ and $\overline{\text{MCAS}}$. Used for write commands.

High-impedance—$\overline{\text{MWE}}$ is always driven unless the memory controller is idle."

9.3.2.2, 9-8 — In Table 9-4, correct the "MCKE" signal name to read "MCKE[0:1]".

Also, change the first full sentence of the MCKE[0:1] signal description to read:

"Two identical output signals (each hereafter referred to simply as MCKE) used as the clock enable to one or more SDRAMs."

9.3.2.2, 9-8 — In Table 9-4, change the Timing description for the MCKE[0:1] signal to read:

"Assertion/Negation—Similar timing to MA*n*.

High-impedance—Always driven."

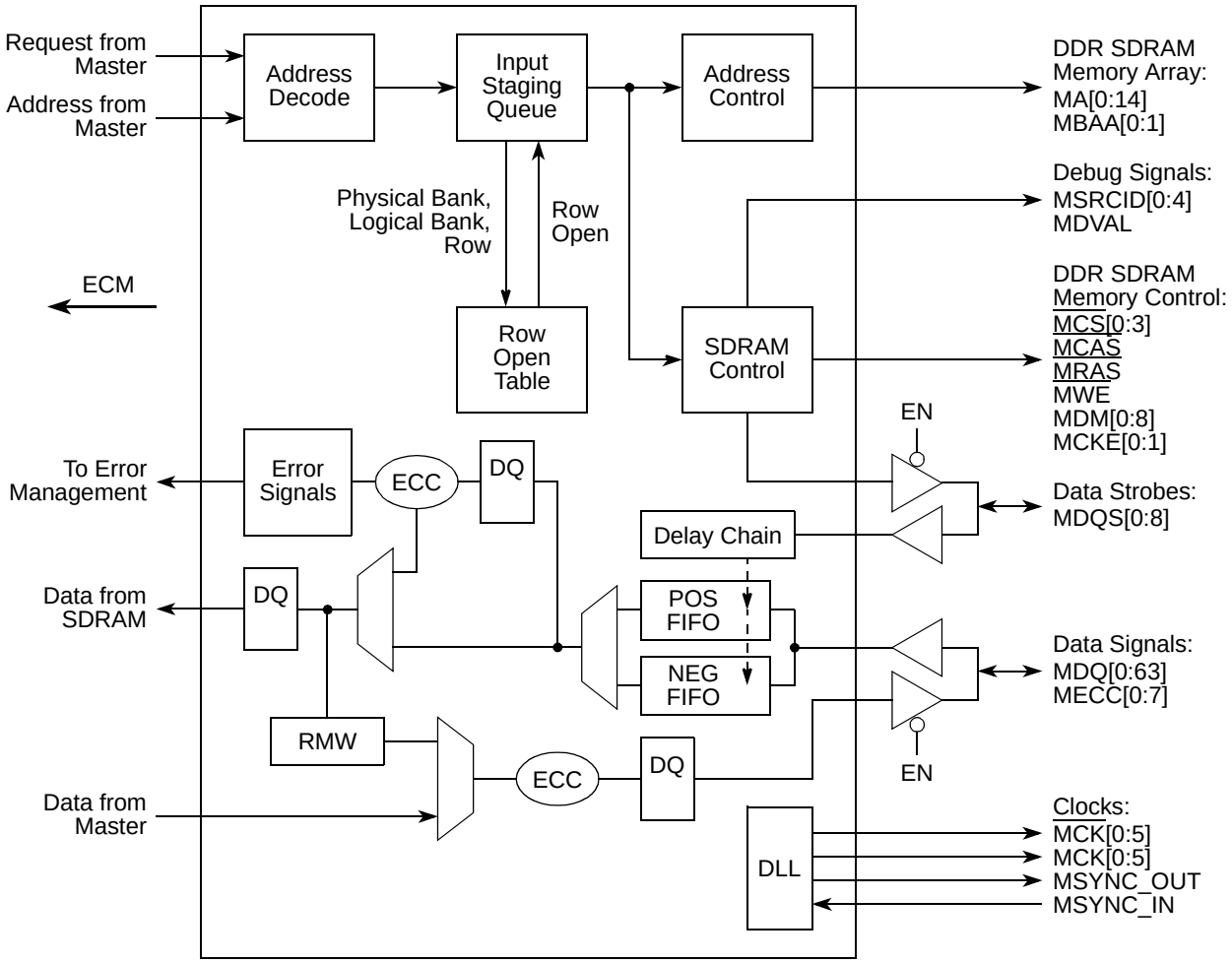
9.4.1.1, 9-10	In Table 9-6, change the CS*n*_BNDS[EA*n*] field description to read: "Ending address for chip select (bank) *n*. This value is compared against the 8 msbs of the address."
9.4.1.4, 9-13	In Table 9-9, replace all instances of |CASLAT| with ⌈CASLAT⌉.
9.4.1.5, 9-14	In Table 9-10, change the setting of the MBEE bit in the DDR_SDRAM_CFG[ECC_EN] description to read as follows: "ECC enable. Note that uncorrectable read errors may cause the assertion of core_fault_in, which causes the core to generate a machine check interrupt unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, ERR_DISABLE[MBED] must be zero and ERR_INT_EN[MBEE] and ECC_EN must be one to ensure an interrupt is generated. See Section 6.10.2, "Hardware Implementation-Dependent Register 1 (HID1)." 0 No ECC errors are reported. No ECC interrupts are generated. 1 ECC is enabled."
9.4.1.7, 9-16	In Table 9-12, add the following last sentence to the DDR_SDRAM_INTERVAL[REFINT] field description: "Note that REFINT must be set to a non-zero value in order for the DDR to enter sleep mode. See Section 17.5.1.5.3, "Sleep Mode," for additional details."
9.4.1.15, 9-21	Change the setting of MBEE bit in the ERR_DISABLE[MBED] description to read as follows: Multiple-bit ECC error disable 0 Multiple-bit ECC errors are detected if DDR_SDRAM_CFG[ECC_EN] is set. They are reported if ERR_INT_EN[MBEE] is set. Note that uncorrectable read errors cause the assertion of *core_fault_in*, which causes the core to generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, MBED must be zero and ERR_INT_EN[MBEE] and ECC_EN must be one to ensure that an interrupt is generated. 1 Multiple-bit ECC errors are not detected or reported.
9.4.1.16, 9-21	Change the setting of the MBEE bit in the ERR_INT_EN[MBEE] description to read as follows: "Multiple-bit ECC error interrupt enable. Note that uncorrectable read errors may cause the assertion of *core_fault_in*, which causes the core to

generate a machine check interrupt, unless it is disabled (by clearing HID1[RFXE]). If RFXE is zero and this error occurs, ERR_DISABLE[MBED] must be zero and MBEE and DDR_SDRAM_CFG[ECC_EN] must be set to ensure that an interrupt is generated. For more information, see Section 6.10.2, “Hardware Implementation-Dependent Register 1 (HID1).”

0 Multiple-bit ECC errors cannot generate interrupts.

1 Multiple-bit ECC errors generate interrupts.”

9.5, 9-25 Replace Figure 9-21 with the figure below:



9.5, 9-25 Remove the last sentence of the fourth paragraph.

9.5.1.1, 9-30 In the first two columns of Table 9-26, change “Mbytes” to “Mbits” and “Gbytes” to “Gbits”.

9.5.1.1, 9-30 Delete the second-to-last paragraph of this sentence that starts with the words, “If a disabled bank...”

9.5.4–9.5.7, 9-35–9-39

In all burst examples, re-number the data beats in the figures to be D0, D1, D2, D3, D0, D1, D2, D3.

9.5.4, 9-35

In the last paragraph, change the phrase, “see Figure 9-26 for a single-beat read operation,” to read “see Figure 9-26 for a back-to-back burst read operation,”.

9.5.4, 9-36

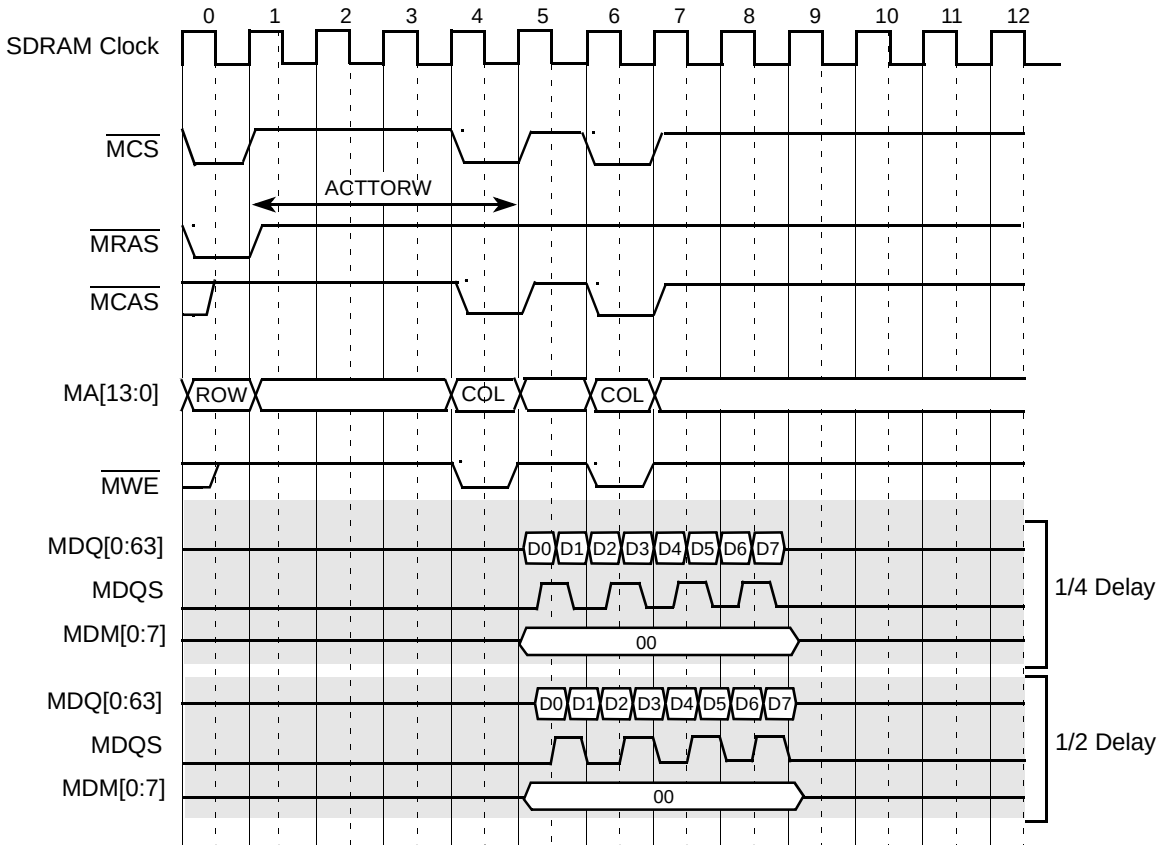
Change the title of Figure 9-28 to “DDR SDRAM Burst Write Timing—ACTTORW = 4”.

9.5.6, 9-38

Change the last sentence of this section to read “Figure 9-31 shows the registered DDR SDRAM DIMM back-to-back burst write timing.”

9.5.7, 9-39

Replace the existing Figure 9-32 with the figure below:



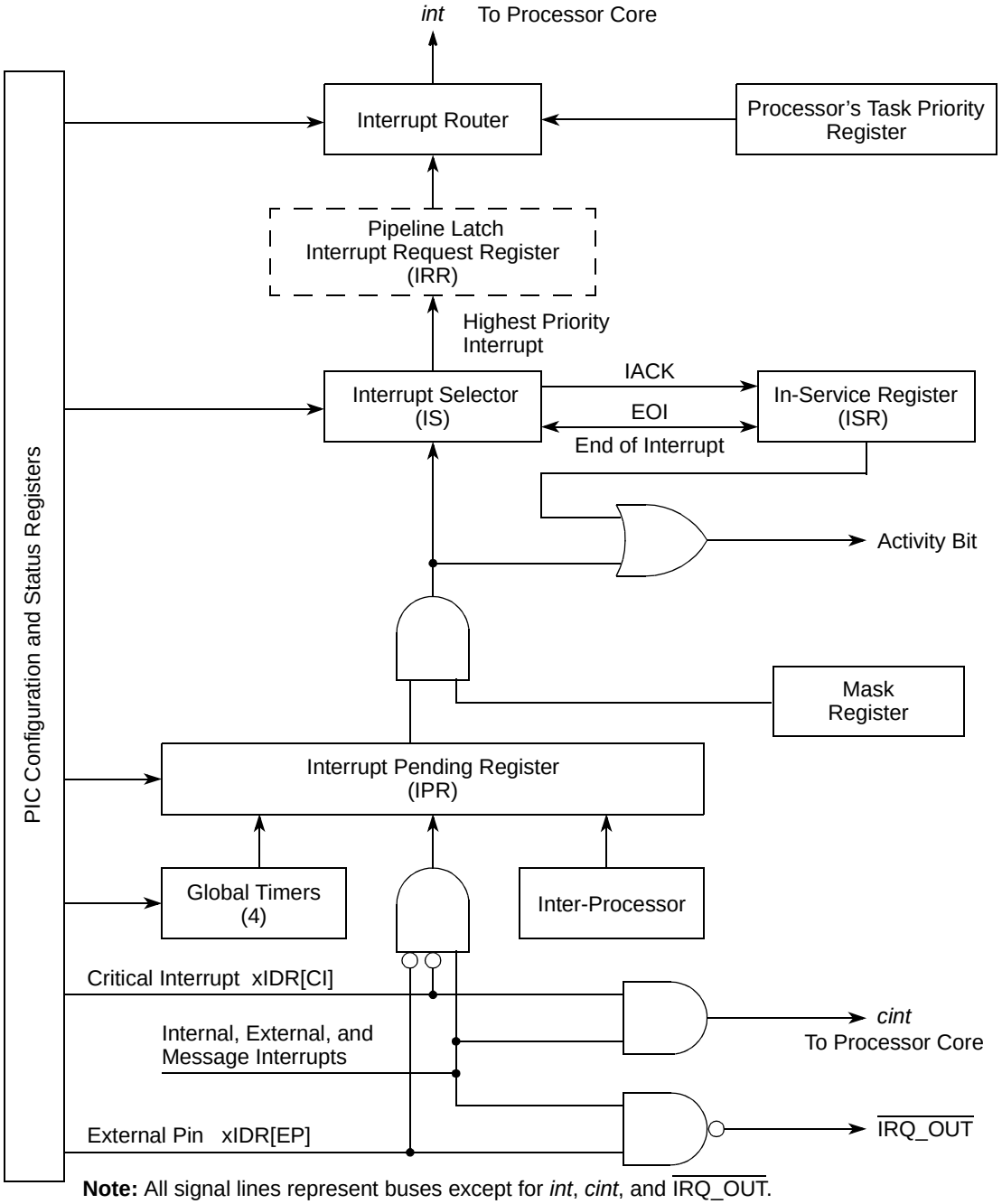
9.5.8, 9-40

In the paragraph preceding Section 9.5.8.1, change the reference to “Two sets of auto refresh commands...” to read “Three sets of auto refresh commands...”. Also, change the reference to “...commands are also staggered in two groups...” to read “... commands are also staggered in three groups...”

9.5.8.1, 9-40	In Figure 9-33, remove the annotation of “0 or 3” in clock 12.
9.5.8.2, 9-41	After the second paragraph, add the following paragraph: “All open pages are precharged before self refresh mode is entered.” Also, correct the last sentence of the fifth paragraph to read: “This mode is controlled with DDR_SDRAM_CFG[DYN_PWR].”
9.5.9, 9-43	Replace the third and fourth sentences of this section with the following: “If ECC is enabled for a sub-doubleword write transaction, a full read-modify-write is performed to properly update ECC bits. If ECC is disabled then no read-modify-write is required for sub-doubleword writes, and the data masks (MDM[0:8]) are used to prevent writing unwanted data to SDRAM.”
9.5.11, 9-44	Add the following sentences before the second-to-last sentence of the second paragraph: “This read-modify-write operation is performed as an atomic transaction in the DDR controller. The write command is then issued 3-5 memory clocks after the completion of the read, depending on various system parameters.”
9.5.12, 9-46	In the last paragraph, delete the three sentences that start and end with, “For all memory select errors...” and “...to show the transaction is not real.”
9.6.1, 9-48	In the second sentence, replace “after the DLL has locked” with the following: “after the memory clocks are stable (that is, the DLL has locked or initialization is complete of all clock related configuration registers).” Replace the last sentence of the first paragraph with the following: “After MEM_EN has been set, the DDR memory controller automatically performs the JEDEC-compliant initialization sequence to initialize memories according to the information in the SDMODE and ESDMODE fields of the DDR_SDRAM_MODE register. The initialization sequence is as follows:” After the numbered list, add the following sentence: “Note that the BA0 and BA1 bits are automatically driven appropriately during the MODE REGISTER SET commands.” Then the final sentence should be changed to, “After this automatic initialization is complete the memory array is ready for access and the memory controller begins processing memory transactions as they arrive.”

10.2.2, 10-8	In Table 10-5, delete the last sentence in the Timing/Negation section for IRQ[0:11] signals about edge-sensitive interrupts.
10.3.5.3, 10-30	Append the last sentence of the first paragraph to read: “Reading the corresponding message register or writing a 1 to a status bit clears the corresponding message interrupt and the status bit.”
10.3.7, 10-37	In Table 10-35, correct the who am I register name from “WHOAMI0” to “WHOAMI.”
10.4.1, 10-42	Add the following new paragraph after the first paragraph: “Note that the IPR, IS, and IRR are internal registers that are not accessible to the programmer.”

10.4.1, 10-42 Replace the existing Figure 10-37, with the figure below:



Note: All signal lines represent buses except for *int*, *cint*, and $\overline{\text{IRQ\_OUT}}$.
The behavior of the PIC unit is not defined if both the EP and CI bits of the same interrupt destination register are set.

11.3, 11-4 In Table 11-3, correct the I2CDFSRR reset value to be 0x10.

11.4.5.2, 11-18	Insert the following two sentences after the eighth sentence of the first paragraph: "The boot sequencer expects the address offset to be a 32-bit (word) offset, that is, the 2 low-order bits are not included in the boot sequencer command. For example, to access LAWBAR0 (byte offset of 0x00C08), the boot sequencer ADDR[0:17] should be set to 0x00302."
11.4.5.2, 11-18	Insert the following sentence after the polynomial used for CRC calculation in the fourth paragraph: "CRC values are calculated using the above polynomial with a start value of 0xFFFF_FFFF and an XOR with 0x0000_0000."
Chapter 12	Correct all instances of "UPWAIT" to "LUPWAIT."
12.1.3.1, 12-3	Change the third sentence of the first paragraph to read: "In addition to establishing the frequency of the external local bus clock, CLKDIV also affects the resolution of signal timing shifts in GPCM mode, and the interpretation of UPM array words in UPM mode."
12.1.3.1, 12-4	In the last sentence, change "LCLK[0:3]" to "LCLK[0:2]."
12.2, 12-5	Delete the Reset State (outputs) column (and references to it) of Table 12-1.
12.2, 12-6, 12-7, and 12-9	In Table 12-2, change all instances of "RAS" to "$\overline{\text{RAS}}$" and all instances of CAS to $\overline{\text{CAS}}$.
12.3.1.2.3, 12-16 and 17	In Table 12-7, Table 12-8, Figure 12-4, and Figure 12-5, change bits 17–18 to reserved.
12.4.4, 12-61	Insert the following footnote associated with the word 'signals' in the second sentence of the section: "If the LGPL4/LGTA/LUPWAIT/LPBSE signal is used as both an input and an output, a weak pullup is required. Refer to the hardware specification for details regarding termination options."
12.4.4.2, 12-65	Add the following third paragraph: "Note that the UPM memory region must be cache-inhibited or write-through (the MMU page must have the I or W bit set) during the time that the UPM array is being written. If the memory is to be cacheable and/or copyback, the MMU must be set accordingly after the UPM array is initialized."
12.4.7.3, 12-55	In the section title, change "CAS" to "$\overline{\text{CAS}}$".

Chapter 13	For consistency, change all instances of “HFE”, “Huge_Frame”, and “HUGE FRAME” to “Huge Frame”.
13.2, 13-7	In the first bullet item of the TSEC features list, change “820.3x” to “802.3x.”
13.2, 13-7	Change the fifth element of the second bullet item in the TSEC features list to read: “– 10/100 Mbps RMII”
13.5.2, 13-14	In Table 13-3, add the following as the first row under the TSEC1 FIFO Control and Status Registers section:

0x2_404C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	13.5.3.2.1/13-30

13.5.2, 13-15	In Table 13-3, correct the reset value for the OSTBD register to 0x0800_0000.
13.5.2, 13-16	In Table 13-3, correct the MACCFG1 register from having R/W, R access to R/W access.
13.5.2, 13-16	In Table 13-3, correct the IFSTAT register from having R access to R/W access.
13.5.2, 13-16	In Table 13-3, change the reset value for the TSEC IFSTAT register to be 0x0000_0000.
13.5.2, 13-18	In Table 13-3, correct the CAR1 and CAR2 registers from having R access to R/W access.
13.5.2, 13-18	In Table 13-3, remove the incorrect "Cleared on read" footnote designation from the CAR1 and CAR2 registers.
13.5.3.1.1, 13-20	In Table 13-4, correct the IEVENT[RXC] bitfield description to read as follows: “Receive control interrupt. A control frame was received. If MACCFG1[Rx_Flow] is set, a pause operation is performed lasting for the duration specified in the received pause control frame and beginning when the frame was received. 0Control frame not received 1 Control frame received”
13.5.3.1.1, 13-20	Replace the last two sentences of the first paragraph with: “Clearing the IEVENT bit clears the interrupt signal. The bit in the IEVENT register is cleared if a 1 is written to that bit position. A write of 0 has no effect.”

13.5.3.1.1, 13-21 In Table 13-4, replace the EBERR field description with:

"Ethernet bus error. This bit indicates that a system bus error for a memory read ocurred while a DMA transaction was underway. If the EBERR is set while transmission is in progress, the DMA stops sending data to the Tx FIFO which eventually causes an underrun error (XFUN) and TSTAT[THLT] is set. If the EBERR is set while receiving a frame, the DMA discards the frame and RSTAT[QHLT] is set.

0 No system bus error occurred.

1 System bus error occurred."

13.5.3.1.2, 13-22 Replace the last sentence in the first paragraph with:

"The interrupt signal can be cleared by clearing the corresponding IEVENT bit."

13.5.3.1.3, 13-24 Undocument bitfields TXEDIS, CRL/XDADIS, and XFUNDIS.

13.5.3.1.7, 13-27 and 28

Remove bitfield DMACTRL[TOD] from Figure 13-12 and Table 13-10. This bit position (29) is now reserved.

13.5.3.1.7, 13-28 In Table 13-10, add the following paragraph after the second paragraph of the WOP field description:

"A buffer descriptor is considered not ready when its TxBD[Ready] field is clear or its TxBD[Data Length] field is zero. It is considered ready when both TxBD[Ready] is set and TxBD[Data Length] is non-zero."

13.5.3.1.7, 13-28 In Table 13-10, change the second sentence of the third paragraph (which becomes the fourth paragraph after incorporating of the above errata) of the WOP field description to read:

"If the TxBD becomes ready, TSEC continues to process the frame."

13.5.3.2.1, 13-30 Insert the following section as the first subsection of Section 13.5.3.2:

13.5.3.2.1 FIFO Pause Control Register (FIFO_PAUSE_CTRL)

FIFO_PAUSE_CTRL, shown in Figure 13-20, is writable by the user to configure the properties of the TSEC FIFO.

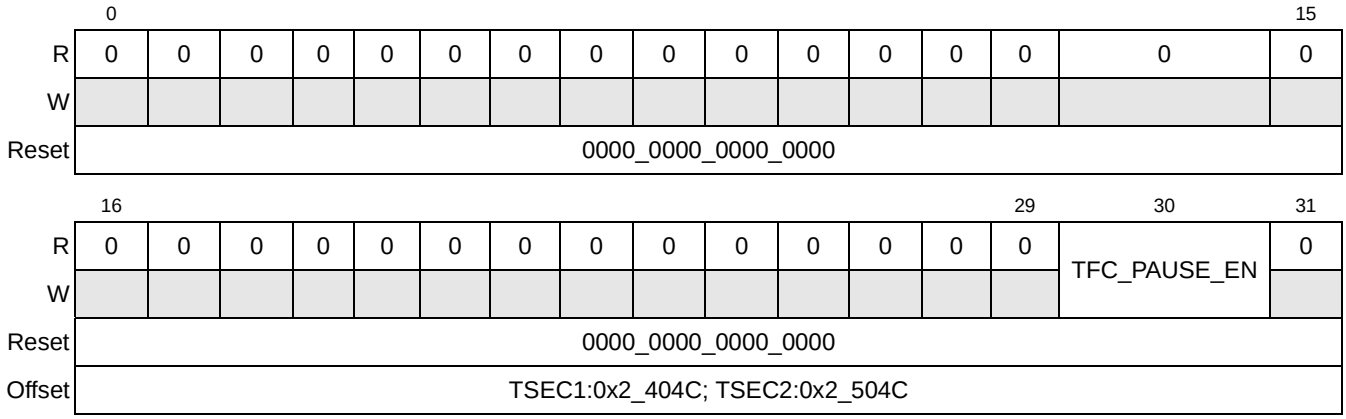


Figure 13-14 FIFO_PAUSE_CTRL Register Definition

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

Table 13-12 FIFO_PAUSE_CTRL Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30	TFC_PAUSE_EN	TFC_PAUSE enable. This bit enables the ability to transmit a pause control frame by setting the TCTRL[TFC_PAUSE] bit. This bit is cleared at reset. 0 Pause control frame transmission disabled. 1 Pause control frame transmission enabled.
31	—	Reserved

13.5.3.3.1, 13-32

In Table 13-15, replace the bitfield description of TCTRL[TFC_PAUSE] with the following:

“Transmit flow control pause frame. Use this bit to transmit a PAUSE frame. To transmit a flow control pause frame, first set FIFO_PAUSE_CTRL[TFC_PAUSE_EN]. Next, set MACCFG1[GTS]. If TFC_PAUSE is then set, the MAC stops transmission of data frames after the current transmission completes. 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 No outstanding pause frame transmission request.

1 Pause frame transmission requested.”

13.5.3.3.6, 13-35 Change the last sentence in the first paragraph to read:

"Although not necessary in most applications, the user can modify this register when the transmitter has been gracefully stopped or halted, as indicated by IEVENT[GTSC] or TSTAT[THLT]."

13.5.3.3.8, 13-38 In Table 13-22, add the following sentence to the OSTBDLEN field description:

"This field must be greater than zero in order for a transfer to take place."

13.5.3.4.2, 13-40 In Table 13-25, change the description for the RSTAT[QHLT] field to read:

"RxBD queue is halted. When IEVENT[BSY] or IEVENT[EBERR] is set during reception of a packet, RSTAT[QHLT] is also set. In order to begin receiving packets again, the user must clear RSTAT[QHLT]. This bit is set whenever the TSEC reads an RxBD with its Empty field cleared or encounters a system bus error while processing an RX packet. It is a hardware-initiated stop indication (DMA_CTRL[GRS] being set by the user does not cause this bit to be set.). The current frame and all other frames directed to the halted queue are discarded. A write with a value of 1 re-enables the queue for receiving.

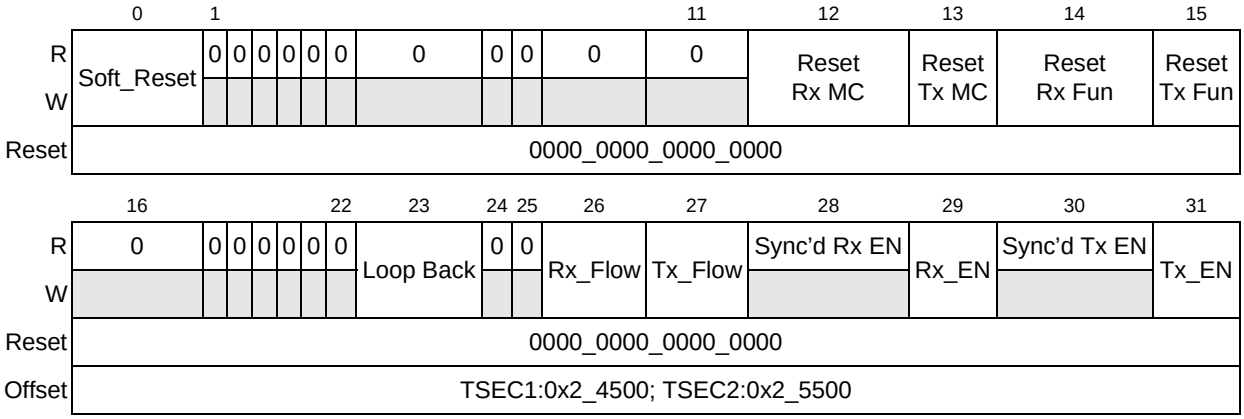
0 RxBD queue is enabled for Ethernet reception. (That is, it is not halted.)

1 All Ethernet controller receive activity to RxBD queue is halted."

13.5.3.4.2, 13-39 Replace Figure 13-27 with the following figure:

	0							7	8	9																					31
R	0	0	0	0	0	0	0	0	QHLT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																															
Reset	0000_0000_0000_0000_0000_0000_0000_0000																														
Offset	TSEC1:0x2_4304; TSEC2:0x2_5304																														

13.5.3.6.1, 13-47 Replace Figure 13-34 with the following figure:



13.5.3.6.1, 13-48 In Table 13-32, add the following sentence to the MACCFG1[Rx_EN] field description:

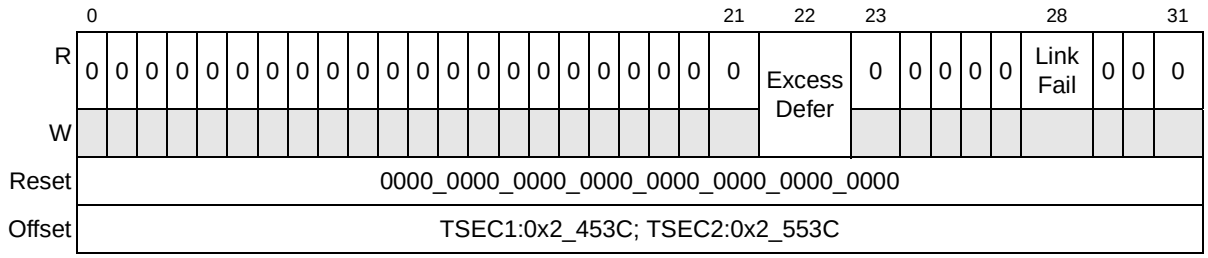
"If set, prior to clearing this bit, set DMACTRL[GRS] then confirm subsequent occurrence of the graceful receive stop interrupt (IEVENT[GRSC] is set)."

13.5.3.6.1, 13-48 In Table 13-32, add the following sentence to the MACCFG1[Tx_EN] field description:

"If set, prior to clearing this bit, set DMACTRL[GTS] then confirm subsequent occurrence of the graceful transmit stop interrupt (IEVENT[GTSC] is set)."

13.5.3.6.6, 13-52 In Table 13-37, clarify the bit description of the Mgmt Clock Select field of the MIIMCFG register so that instead of concluding with "... divided by eight." the third sentence concludes with "... divided first by eight and then futher divided by the following value:".

13.5.3.6.12, 13-55 Delete the first sentence and replace Figure 13-45 with the following figure (exposing the Link Fail status field):



13.5.3.6.12, 13-56 In Table 13-43, replace the last row with the following rows:

23–27	—	Reserved
28	Link Fail	Link Fail. This bit indicates the status of signal detection. 0 The100X module has detected a “signal detect” for longer than 330 mS. 1 The100X module has detected a “signal detect” for less than 330 mS or not at all.
29–31	—	Reserved

13.5.3.6.12, 13-55 In Figure 13-45, correct the reset value for the IFSTAT register to be 0x0000_0000.

13.5.3.7.44, 13-79 Correct the first sentence to read:
"Carry register bits are cleared when written with a one."

13.5.3.7.45, 13-80 Correct the first sentence to read:
"Carry register bits are cleared when written with a one."

13.5.3.7.47, 13-83 In Figure 13-94, change bit 30 to be “Reserved”.

13.5.3.8.1, 13-84 Change the third sentence of the first paragraph to read:

“When the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 high-order bits (0–7) of the CRC remainder are mapped to one of the 256 entries.”

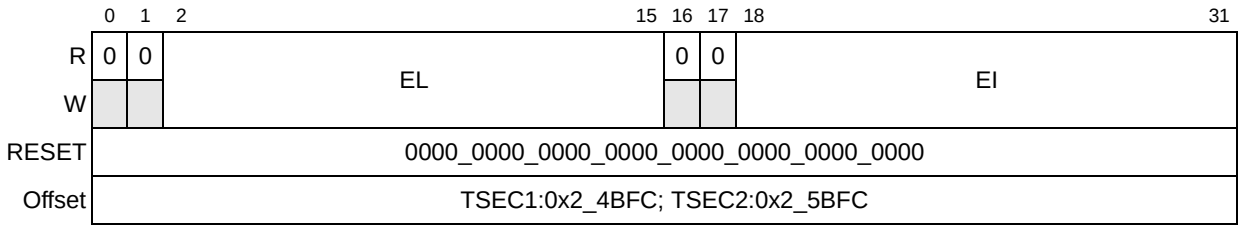
13.5.3.8.1, 13-85 In Table 13-93, add the following sentence to the IADDR*n* field description:

“For instance, the MSB of IADDR0 correlates to entry 0 and the LSB of IADDR7 correlates to entry 255.”

13.5.3.8.2, 13-85 In Table 13-94, add the following sentence to the GADDR*n* field description:

“For instance, the MSB of GADDR0 correlates to entry 0 and the LSB of GADDR7 correlates to entry 255.”

13.5.3.9.2, 13-87 Replace Figure 13-98 with the following figure to correspond to Table 13-96:



13.6.2.2, 13-108	Replace the existing procedure list with the following list: “Following is a procedure to gracefully reset and reconfigure the MAC: 1. Set GTS bit in DMACTRL register 2. Poll GTSC bit in IEVENT register until detected as set 3. Clear both Rx_EN and Tx_EN bits in MACCFG1 4. Set GRS bit in DMACTRL register 5. Poll GRSC bit in IEVENT register until detected as set 6. Set Soft_Reset bit in MACCFG1 register 7. Clear Soft_Reset bit in MACCFG1 register 8. Load TBASE with new TxBD pointer 9. Load RBASE with new RxBD pointer 10. Set up other MAC registers (MACCFG1, MAXFRM, and so on) 11. Set WWR, WOP, TOD bits in DMACTRL register 12. Clear THLT bit in TSTAT register and QHLT bit in RSTAT register by writing 1 to these bits 13. Clear GRS/GTS bits in DMACTRL (do not change other bits) 14. Enable Tx_EN/Rx_EN in MACCFG1 register”
13.6.2.2, 13-108	Insert the following step between steps 3 and (formerly) 4: "4. Wait for a period of 9.6 Kbytes worth of data on the interface (~8ms worst case)."
13.6.2.2, 13-109	Correct step 11 of the graceful reset procedure to read: "11. Set WWR and WOP bits in DMACTRL register"
13.6.2.3, 13-109	Replace the second paragraph with the following: "If the user has a frame ready to transmit, a transmit-on-demand function may be emulated while in polling mode by using the graceful-transmit-stop feature. First, clear the IMASK[GTSCEN] bit to mask the graceful-transmit-stop complete interrupt. Next set, then immediately clear the DMACTRL[GTS] bit. Clear the resulting IEVENT[GTSC] bit. Finally, the IMASK[GTSCEN] bit may be set once again."
13.6.2.3, 13-110	Insert the following sentence between the third and (formerly) fourth sentences of the sixth paragraph: "The pause duration is defined by the received pause control frame and begins when the frame was first received."

13.6.2.6.2, 13-114 In the first paragraph, replace the third and fourth sentences with the following:

"The eight high-order bits of a cyclic redundancy check (CRC) checksum are used to index into the hash table."

13.6.2.6.3, 13-114 Add the following section after Section 13.6.2.6.2:

13.6.2.6.3CRC Computation Examples

There are many algorithms for calculating the CRC value of a number. Refer to the RFC 3309 standard, which can be found at http://www.faqs.org/rfcs/rfc3309.html, to compute the CRC value for the purposes of TSEC. The RFC 3309 algorithm uses the following polynomial to calculate the CRC value:
x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x1+x0 or 0x04c11db7.

Given a destination MAC address of DA=01000CCCCCCC, the algorithm results in a CRC remainder value of 0xA29F4BBC.

Bit-reversing the low-order byte of the CRC value (0xBC) yields BR_CRC = 0x3D = 0b00111101

The high-order 3-bits of the new BR_CRC value are used to select which 32-bit register (of the 8) to use. This example maps the DA to register 1.

High-order 3 bits of BR_CRC: HO_CRC = 0b001 = 1

The low-order 5 bits are used to select which bit to set in the given register (with a value of 0 setting 0x8000_0000 and 31 setting 0x0000_0001). Therefore, the example DA maps to bit 29 of register 1.

Low-order 5 bits of BR_CRC: LO_CRC = 0b11101 = 29

Therefore, GADDR1 is ORed with the value 0x0000_0004.

Additional calculated examples follow:

Example 1:

- Destination MAC address: DA = 01005E000128
- CRC remainder value: CRC = 0x821D6CD3
- Bit-reversed least-significant byte of CRC value: BR_CRC = 0xCB = 0b11001011
- High-order 3 bits of BR_CRC: HO_CRC = 0b110 = 6

- Low-order 5 bits of BR_CRC: LO_CRC = 0b01011 = 11
- GADDR6 = 0x0010_0000

Example 2:

- Destination MAC address: DA = 0004F0604F10
- CRC remainder value: CRC = 0x1F5A66B5
- Bit-reversed least-significant byte of CRC value: BR_CRC = 0xAD = 0b10101101
- High-order 3 bits of BR_CRC: HO_CRC = 0b101 = 5
- Low-order 5 bits of BR_CRC: LO_CRC = 0b01101 = 13
- GADDR5 = 0x0004_0000

13.6.2.8, 13-116 — In Table 13-115, corrected the RXC description to read as follows:

"Receive control: A control frame was received. As soon as the transmitter finishes sending the current frame, a pause operation is performed lasting for the duration specified in the received pause control frame and beginning when the frame was first received."

13.6.2.11, 13-118 — Insert the following paragraph after the first paragraph:

"Programming note: When the TSEC encounters a halt condition (TSTAT[THLT] is set), it stops processing the frame at the current TxBD. The TSEC relies on the user to manage the buffer descriptor pointer, TBPTR, or the buffer descriptor queue before resuming transmissions. Once the TSEC resumes, it fetches the TxBD pointed to by TBPTR."

13.6.3, 13-120 — Between the second and third paragraph, add the following:
"The status field of the BD is 16-bit field, as is the length field. The data buffer pointer is a 32-bit field. Therefore, the BDs should be accessed with the following C structure:

```
typedef unsigned short uint_16; /* choose 16-bit native type */
typedef unsigned int uint_32; /* choose 32-bit native type */
typedef struct bd_struct {
    uint_16 flags;
    uint_16 length;
    uint_32 bufptr;
};"
```

13.6.3.2, 13-124 — In Table 13-120, change the LG bitfield description to read as follows:

"Rx frame length violation. Written by TSEC. (Only valid if L is set.) A frame length greater than maximum frame length was recognized while MACCFG2[HUGE FRAME] was set. Note, if MACCFG2[HUGE FRAME] is cleared, the frame is truncated to the value programmed in the Maximum Frame Length register."

13.6.4, 13-125 — Change the first paragraph to read:

"Some applications require the ability to identify selected portions of data within a frame's data payload. This process is called extraction, although the data is not truly removed. Rather than literarily extracting a section of the data and copying it into a new memory location, the data is placed in the L2 cache. This allows the processor to quickly access critical frame information as soon as the processor is ready without having to first fetch the data from main memory. This results in substantial improvement in throughput and hence reduction in latency."

13.7.1.1, 13-129 — In Table 13-123, add the following step immediately following the optional DMACTRL initialization step:

Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]

13.7.1.2, 13-129 — Change the title of Table 13-124 to read "GMII Interface Mode Signal Configuration."

13.7.1.2, 13-129 — In Table 13-124, remove the TX_CLK signal from the list of GMII interface signals and reduce the total number of signals listed at the bottom to 22.

13.7.1.2, 13-132 — In Table 13-126, add the following step immediately following the optional DMACTRL initialization step:

Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]

13.7.1.3, 13-136 — In Table 13-129, add the following step immediately following the optional DMACTRL initialization step:

Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]

13.7.1.4, 13-137 — Change the title of Table 13–130 to read "RGMII Interface Mode Signal Configuration."

13.7.1.4, 13-140 — In Table 13-132, add the following step immediately following the optional DMACTRL initialization step:

Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]

13.7.1.5, 13-144 — In Table 13-135, add the following step immediately following the optional DMACTRL initialization step:

Initialize FIFO_PAUSE_CTRL, FIFO_PAUSE_CTRL[0000_0000_0000_0000_0000_0000_0000_0010]

Chapter 14 — Correct the DMA signal enumeration for consistency as in the following example:

"DMA0_DREQ" to "DMA_DREQ0"

14.2.2, 14-5 and

14.4.1.3, 14-32 — In the third bullet of the list of signals defined for the external control interface, clarify that the falling edge of $\overline{\text{DMA\_DREQ}}$ sets MR*n*[CS] and for $\overline{\text{DMA\_DDONE}}$ with the following:
"$\overline{\text{DMA\_DDONE}}$ assertion indicates that the DMA engine has completed the transfer. SR*n*[CB] is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface."

14.3.1, 14-7 — In Table 14-4, designate the following memory locations as "Reserved":

0x2_112C

0x2_1148–0x2_117C

0x2_11AC

0x2_11C8–0x2_11FC

0x2_122C

0x2_1248–0x2_127C

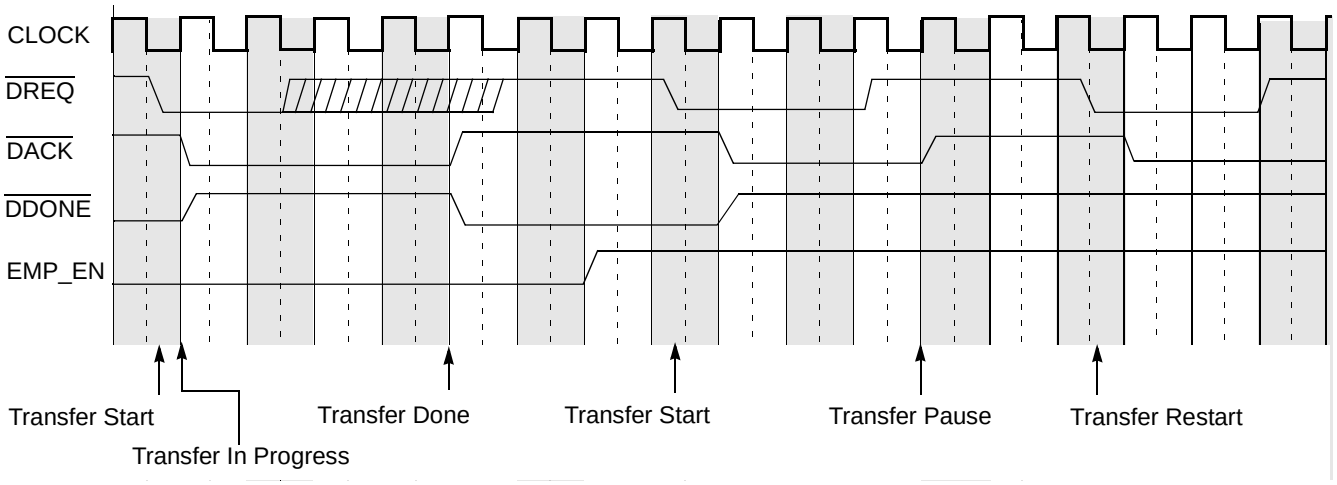
0x2_12AC

0x2_12C8–0x2_12FC

14.3.2.5.1, 14-17 — Change the designation for the source address register when used for RapidIO maintenance transactions from "SARM*n*" to "SAR*n* for RapidIO maintenance transactions," reflecting the fact that any single SAR*n* register can assume one of two formats.

14.3.2.7.1, 14-20 — Change the designation for the destination address register when used for RapidIO maintenance transactions from "DARM*n*" to "DAR*n* for RapidIO

	maintenance transactions," reflecting the fact that any single DAR*n* register can assume one of two formats.
14.3.2.10, 15-22	In Figure14-17, correct the offset value for CLSDAR2 to 0x2_1234.
14.3.2.11, 15-23	In Figure14-18, correct the offset value for the NLSDAR*n*–DMA 2 next list descriptor address register to 0x2_123C.
14.5.1.1.1, 14-27	Change step 1 in the sequence to read, "Poll the channel state (see Table 14-21 "Channel State Table") to confirm that the specific DMA channel is idle."
14.5.1.1.2, 14-28	Change step 1 in the sequence to read, "Poll the channel state (see Table 14-21 "Channel State Table") to confirm that the specific DMA channel is idle."
14.5.1.1.3, 14-29	Change step 2 in the sequence to read, "Poll the channel state (see Table 14-21 "Channel State Table") to confirm that the specific DMA channel is idle."
14.5.1.1.4, 14-29	Change step 3 in the sequence to read, "Poll the channel state (see Table 14-21 "Channel State Table") to confirm that the specific DMA channel is idle."
14.5.1.2.3, 14-30	Change step 2 in the sequence to read, "Poll the channel state (see Table 14-21 "Channel State Table") to confirm that the specific DMA channel is idle."
14.5.1.2.4, 14-31	Change step 3 in the sequence to read, "Poll the channel state (see Table 14-21 "Channel State Table") to confirm that the specific DMA channel is idle."
14.4.1.3, 14-32	Replace Figure 14-22 with the following, showing more flexible negation of $\overline{\text{DREQ}}$:



14.5.1.3, 14-42	Remove the last sentence of the second paragraph.
15.2, 15-12	In Table 15-2, add the following to the timing description for the $\overline{\text{PCI\_PERR}}$ signal: "**Note:** If a parity error occurs on the last data beat of a PCI-X transaction with inbound data (outbound read with split completion data or inbound write), the MPC8560 asserts PERR longer than permitted by the PCI-X specification. This condition may cause a subsequent transaction to erroneously detect a parity error. The condition may be remedied by placing a stronger pull-up resistors on the PERR signal. For example: •1 Kohm for 133 MHz point-to-point operation •2 Kohm for 66 MHz operation •4 Kohm for 33 MHz operation. These values are stronger than the specification allows (5 Kohm minimum for pull-up resistors)."
15.3.1.2, 15-20	Add the following sentence to the end of the third paragraph: "Note that outbound translation windows must not overlap the configuration access registers."
15.3.1.2.4, 15-22	In Figure 15–9, add a footnote to EN field (bit 0) that states, " For POWAR0, translation is always enabled. The enable field (EN) may be read and written, but the value is ignored."
15.3.1.2.4, 15-22	In Table 15–10, add the following to the description for bit 0 (EN), "Note that for POWAR0, translation is always enabled. The enable field (EN) may be read and written, but the value is ignored."
15.3.1.2.4, 15–23	In Table 15–10, add the following to the description for bits 26–31 (OWS), "Also note that for POWAR0, setting OWS to less than 4 Gbytes causes addresses that miss in the other outbound windows to be aliased to the smaller address range defined by POWAR0[OWS] and POTAR0."
15.3.1.4, 15–27	Insert the following between the second and third paragraphs: "Note that some errors are reported in two bits—one in the PCI/X error detect register (ERR_DR) and another in the PCI bus status register in the PCI/X configuration header. These bits must be cleared separately; that is, clearing one does not clear the other. For example, clearing the ERR_DR[Mstr abort error] does not clear the received master abort bit in the PCI bus status register. In these cases, both bits must be cleared before further error reporting can occur. Refer to Table 15-57 for PCI mode error actions and Table 15-64 for PCI-X mode error actions. Likewise, some

	errors are enabled by programming two bits—one in the PCI/X error enable register and another in the PCI bus command register in the PCI/X configuration header.”
15.3.1.4, 15–28	Insert the following between the fourth and fifth paragraphs: “If a data parity error occurs during an inbound configuration write access, the error is reported and captured. However, the erroneous data is written to the register specified in the transaction. Therefore, PCI data parity error recovery routines must include reinitialization of the PCI/X configuration register if the error occurred during a configuration write.”
15.3.1.4.7, 15–32	Add the following sentence before Figure 15-21: “Note that for inbound reads that have data parity errors, only the address (ERR_ADDR and ERR_EXT_ADDR) and attributes (ERR_ATTRIB) are captured. The data is not captured.”
15.3.1.4.7, 15–32	Add the following immediately before Figure 15-21: “Also note that PCI-X split completion error messages for outbound DWORD writes are incorrectly reported in the PCI/X error data low capture register (ERR_DL).”
15.3.1.4.8, 15-33	Add the following sentence before Figure 15-22: “Note that for inbound reads that have data parity errors, only the address (ERR_ADDR and ERR_EXT_ADDR) and attributes (ERR_ATTRIB) are captured. The data is not captured.”
15.3.2, 15-35	Insert the following after Figure 15-26: “Note that software must be restricted from accessing any reserved or undefined register space on the MPC8560; doing so may hang the device.”
15.3.2.3, 15-37	In Table 15-27, add the following sentence to the description of the Bus master bit field: “Note that the Bus master bit in the MPC8560’s PCI bus command register must be set before any outbound configuration access is attempted.”
15.3.2.4, 15–37	Insert the following between the first and second paragraphs: “Note that some errors are reported in two bits—one in the PCI bus status register and another in the PCI/X error detect register (ERR_DR). These bits must be cleared separately; that is, clearing one does not clear the other. For example, clearing the ERR_DR[Mstr abort error] does not clear the received master abort bit in the PCI bus status register. In these cases, both bits must be cleared before further error reporting can occur. Refer to Table 15-57 for PCI mode error actions and Table 15-64 for PCI-X mode

error actions. Likewise, some errors are enabled by programming two bits—one in the PCI bus command register and another in the PCI/X error enable register (ERR_EN).”

15.3.2.9, 15-40 In Figure 15-35, change the reset value from “0000_0000” to “0000_0000 (PCI); 0000_1000 (PCI-X)”.

15.3.2.19, 15-47 In Table 15-46, change the description for bit 0 to read as follows:

“PCI agent/host. Read-only. Indicates the reset value of the PCI host/agent configuration signal, $\overline{\text{LWE3}}$.

0 PCI interface is in host mode

1 PCI interface is in agent mode”

15.4.1, 15-51 Replace the second and third sentences of the third paragraph with the following:

“The state of the reset signal is reflected in bit 15 (read-only) of the PCI bus arbitration control register (PBACR[PAD]). Note that PAD is the inverse of the arbiter configuration signal; that is, when PAD = 0 the arbiter is enabled, and when PAD = 1 the arbiter is disabled.”

15.4.1.3, 15-53 Add the following to the end of the first paragraph:

“Note that the PCI-X bus arbiter does not support the broken master lockout function.”

15.4.2.11.1, 15-69 Insert the following between the first and second sentences in the first paragraph:

“Note that the Bus Master bit in the MPC8560’s PCI bus command register must be set before an outbound configuration access is attempted.”

15.4.2.11.2–15.4.2.11.3, 15-69–15-71

Replace the existing sections with the following:

15.4.2.11.2 Accessing the PCI Configuration Space in Host Mode

To access the configuration space, a 32-bit value must be written to the PCI CFG_ADDR register that specifies the target PCI bus, the target device on that bus, and the configuration register to be accessed within that device. Note that the Bus Master bit in the MPC8560’s PCI bus command register must be set before an outbound configuration access is attempted. Device 0 on PCI/X bus 0 is the MPC8560 itself; thus, device 0, bus 0 is used to access the internal PCI/X configuration header.

When the MPC8560 detects an access to PCI CFG_DATA, it checks the enable flag and the device number in the PCI 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. 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. The device number 0b1_1111 is used for performing interrupt-acknowledge and special-cycle transactions. See Section 16.4.2.12, “Other Bus Transactions,” for more information.

See Section 16.3.1.1, “PCI/X Configuration Address Register (CFG_ADDR),” for details on PCI CFG_ADDR and Section 16.3.1.1.2, “PCI/X Configuration Data Register (CFG_DATA),” for details on PCI CFG_DATA.

Note that because all PCI/X registers are intrinsically little-endian, in the following examples, the data in the configuration register is shown in little-endian order. PowerPC processor accesses to the PCI 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 byte lane redirection from the little-endian PCI bus is performed internally.

Example: Configuration sequence, 4-byte data read from the revision ID/standard programming interface/subclass code/class code registers at address offset 0x08 of the PCI/X configuration header (device 0 on the PCI/X bus 0 is the MPC8560 itself).

```
Initial values:
r0 contains 0x8000_0008
r1 contains CCSRBAR + 0x0_8000 (Address of PCI CFG_ADDR
register)
r2 contains CCSRBAR + 0x0_8004 (Address of PCI CFG_DATA
register)
r3 contains 0xFFFF_FFFF
Register at 0x08 contains 0x0B20_0002 (0x0B to 0x08)

Code sequence:
stw r0, 0 (r1)
lwbrx r3, 0 (r2)

Results:
Address CCSRBAR + 0x0_8000 contains 0x8000_0008
Register r3 contains 0x0B20_0002
```

Example: Configuration sequence, 4-byte data write to PCI/X register at address offset 0x14 of Device 1 on PCI/X bus 0.

```
Initial values:
r0 contains 0x8000_0814
r1 contains CCSRBAR + 0x0_8000 (Address of PCI CFG_ADDR
register)
r2 contains CCSRBAR + 0x0_8004 (Address of PCI CFG_DATA
register)
r3 contains 0x1122_3344
Register at 0x14 contains 0xFFFF_FFFF (0x17 to 0x14)

Code sequence:
stw r0, 0 (r1)// Update PCI CFG_ADDR register to point to
    //register offset 0x14 of device 1.
stwbrx r3, 0 (r2)

Results:
Address CCSRBAR + 0x0_8000 contains 0x8000_0814
Register at 0x14 contains 0x1122_3344 (0x17 to 0x14)
```

Example: Configuration sequence, 2-byte data write to PCI register at address offset 0x1C of Device 1 on PCI/X bus 0.

```
Initial values:
r0 contains 0x8000_081C
r1 contains CCSRBAR + 0x0_8000
r2 contains CCSRBAR + 0x0_8004
r3 contains 0xDDCC_BBAA
Register at 0x1C contains 0xFFFF_FFFF (0x1F to 0x1C)

Code sequence:
stw r0, 0 (r1)
sthbrx r3, 0 (r2)

Results:
Address CCSRBAR + 0x0_8000 contains 0x8000_081C
Register at 0x1C contains 0xFFFF_BBAA (0x1F to 0x1C)
```

15.4.2.11.3 PCI Configuration in Agent and Agent Lock Modes

In general, agents should not access the configuration space of other external PCI devices. Configuration of agents is a function usually reserved for the host. When the MPC8560 is in agent mode, it responds to remote host-generated PCI/X configuration cycles. This occurs when a configuration command is decoded along with the IDSEL input signal being asserted. When the MPC8560 is in agent lock mode, it retries all externally-generated PCI/X configuration cycles until the ACL bit in the PCI bus function register (0x44) is set. See Section 16.5.1, “Power-On Reset Configuration Modes,” for more information.

In either agent or agent lock mode, access to the internal PCI/X configuration header by the processor core is handled as described in Section 16.4.2.11.2, “Accessing the PCI Configuration Space in Host Mode,” using device = 0 and bus = 0 in PCI CFG_ADDR to indicate the internal PCI/X header.

15.4.2.13, 15-75 Delete the second and third sentences in the first paragraph.

15.4.2.13.1, 15-75 Delete the last paragraph.

15.4.2.13.2, 15-76 Add the following between the first and second paragraphs:

“Note that some errors are enabled by programming two bits—one in the PCI bus command register and another in the PCI/X error enable register (ERR_EN). Likewise, some errors are reported in two bits—one in the PCI bus status register and another in the PCI/X error detect register (ERR_DR). These bits must be cleared separately; that is, clearing one does not clear the other. For example, clearing the ERR_DR[Mstr abort error] does not clear the received master abort bit in the PCI bus status register. In these cases, both bits must be cleared before further error reporting can occur.”

15.4.3.6.4, 15-88 Add the following new section that describes the PCI-X split completion message:

15.4.3.6.4 Split Completion Messages

For PCI-X split completion transactions that include a split completion error or split completion message, the transaction has a single DWORD data phase with the format shown in Figure 15-78.

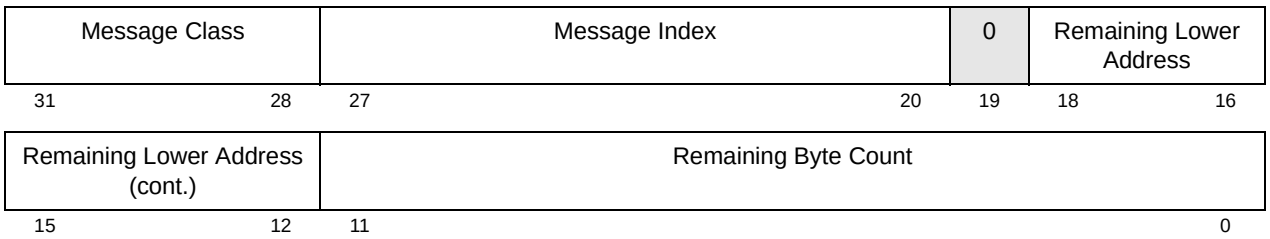


Figure 15-78. PCI-X Split Completion Message Format

Table 15-62 describes the fields of the PCI-X split completion message.

Table 15-62. PCI-X Split Completion Message Summary

AD	Name	Description
31:28	Message Class	A split completion message can be one of the following classes: 0x0 Write completion 0x1 PCI-X bridge error 0x2 Completer error 0x3–0xFReserved
27:20	Message Index	The message index identifies the type of message within the message class. For write completion message class: 0x00 Normal completion For PCI-X bridge error message class: 0x00 Master-abort 0x01 Target-abort 0x02 Write data parity error For completer error message class: 0x00 Byte count out-of-range error 0x01 Split write data parity error 0x8n Device-specific error
19	—	Reserved
18:12	Remaining Lower Address	If the split request was a burst memory read transaction, the remaining lower address is the least significant seven bits of the first byte of data that has not been sent. If the split request was a DWORD transaction, the completer places all zeroes in this field.[1]
11:0	Remaining Byte Count	If the split request was a burst memory read transaction, the remaining byte count is the number of bytes that have not been sent. If the split request was a DWORD transaction, the completer places 0x004 in this field.

[1] Note that the PCI-X 1.0 specification states that if the split request is a DWORD transaction, the completer sets the remaining lower address field (bits 18–12) of the split completion message to zero. However, the MPC8560 deviates from the specification when the split request is a DWORD read transaction to the upper 32 bits of a quad word (that is, the address ends with 1xx); in this case, bit 14 of the split completion message is set to 1.

15.4.3.7, 15-88 Add the following two paragraphs at the beginning of the section:

“PCI-X configuration transactions are similar to PCI configuration transaction as described in Section 16.4.2.11, “Configuration Cycles.” The exceptions are noted in this section.

There is a rare situation where an internal access (originating from either the processor core, or other I/O ports on the device) to the 256-byte PCI-X configuration header may be corrupted by concurrent outbound PCI-X data transactions. The corruption does not occur if the PCI-X configuration transactions and the outbound PCI-X data transactions are not concurrent. The corruption does not occur with inbound PCI-X traffic. This is an issue for any system that mixes internally generated PCI-X configuration transactions with outbound PCI-X data transactions. There are two ways to avoid the situation:

	Ensure that PCI-X configuration transactions are completed before starting PCI-X data transactions. Ensure that all PCI-X outbound data traffic is quiesced before attempting PCI-X configuration transactions. The programmer must ensure all of the following: Non-configuration transactions to PCI-X from the processor core are quiesced. The core is not fetching instructions from PCI-X. DMA transactions to PCI-X are quiesced. RapidIO bridging transactions to PCI-X are quiesced.”
15.4.3.7, 15-88	Add the following before the first sentence: “As is the case for PCI configuration accesses, the Bus master bit in the MPC8560’s PCI bus command register must be set before an outbound configuration access is attempted.”
15.4.3.8.1, 15-91	Add the following between the first and second paragraphs: “Note that some errors are enabled by programming two bits—one in the PCI bus command register and another in the PCI/X error enable register (ERR_EN). Likewise, some errors are reported in two bits—one in the PCI bus status register and another in the PCI/X error detect register (ERR_DR). These bits must be cleared separately; that is, clearing one does not clear the other. For example, clearing the ERR_DR[Mstr abort error] does not clear the received master abort bit in the PCI bus status register. In these cases, both bits must be cleared before further error reporting can occur.”
15.4.3.8.1, 15–93	Add the following after Table 15-64: “When an outbound PCI-X read transaction has been split and the split completion transaction indicates a master-abort or a target-abort, the split-completion error message (SCEM) data field contains information for setting the received master-abort or received target-abort bit in the PCI bus status register (bits 13 and 12, respectively). However, for misaligned reads (that is, reads not aligned on a 64-bit boundary), the error is not reflected in the PCI Status register. The error is reported in the PCI error detection register (ERR_DR) at 0x0_8E00. The master-abort or target-abort can be inferred from the error data capture register (ERR_DL). Drivers that require this information can read the ERR_DR register rather than the PCI bus status register.”

15.5.4, 15–95 Add the following new section:

15.5.4 PCI-X Inbound Reads that Cross Window Boundaries

For Rev. 1 of the MPC8560, inbound PCI-X read requests can cross a window boundary. For Rev. 2, if an inbound PCI-X read request crosses a window boundary, the MPC8560 responds with a split completion message indicating an error but does not return any data. This is a deviation from the PCI-X specification which describes that data is returned up to the window boundary.

15.5.5, 15–95 Add the following new section:

15.5.5 Bridging Between RapidIO and PCI/X

The MPC8560 does not support bridging from RapidIO according to the RapidIO Interoperability Specification. If an external RapidIO device tries to target the MPC8560 with a RapidIO NWRITE-R transaction or with any RapidIO write at a priority level higher than 0, and the RapidIO inbound ATMU routes the transaction to the PCI/PCI-X interface, the MPC8560 could deadlock internally.

Therefore, the use of RapidIO NWRITE-R transactions must be avoided in bridging applications to the PCI/X controller, and all RapidIO writes targeting the PCI/X controller must be of priority level 0. This means transaction ordering is relaxed because read responses will not necessarily push posted writes in the bridge, and writes will never bypass read requests. Note that this specifically precludes the topology where two MPC8560s communicating with each other via PCI or PCI-X may deadlock when two RapidIO devices (one on each MPC8560) are simultaneously performing priority-1 writes to targets on the remote MPC8560 through the PCI/X link between the devices. To support such a topology, RapidIO write transactions that bridge to the other MPC8560 must be priority 0.

16.3, 16-9–2-13 In Table 17-6, correct the access designation for all RapidIO Registers listed as “Special” to “R/W”.

16.3.1.5, 16-16 In Table 17-11, correct the PEFCAR[CTLS] bitfield state description to read as follows:

“0 The RapidIO controller does not support the large common transport route field.

1 The RapidIO controller supports the large common transport route field."

16.3.1.20, 16-30 — In Table 17-26, add the following cross-reference to the field description for PLMREQCSR[C]:

"See Table 17-85 for command symbol formats."

16.4.1, 16-83 — In Table 16-83, add the following to the "Comments" field for "Maintenance Class" reads:
"Although possible to execute, RapidIO maintenance reads from memory locations outside of the RapidIO architectural memory space (0xC_0000–0xC_FFFC) return unreliable data."

16.4.1, 16-86 — In Table 16-85, add the following to the "Comments" field of the symbol type 5 (Link Request) control symbol format description:
"Note that the RapidIO controller causes a checkstop if a software-initiated send training link request command collides with an inbound send training link request which is immediately followed by an inbound training pattern."

17.4, 17-4 — In Table 17-3, change the reset value of the GPINDR register to be 0x*nnnn*_0000.

17.4.1.9, 17-11 — In Figure 17-9, change the reset value of the GPINDR register to be *nnnn_nnnn_nnnn_nnnn*_0000_0000_0000_0000.

17.4.1.14, 17-17 — In Table 17-17, change the bit ranges "32–47" to "0–15" and "48–63" to "16–31".

17.5.1.5.3, 17-23 — Add the following sentence to the end of the first paragraph:

"Note that the DDR controller does not shut down unless DDR_SDRAM_INTERVAL[REFINT] is set to a non-zero value. See Section 9.4.1.7, "DDR SDRAM Interval Configuration (DDR_SDRAM_INTERVAL)," for details."

18.3.1, 18-4 — In Table 18-1, correct the name of the register with offset 0xE_1094 from PMLCB5 to PMLCB8.

19.2.1, 19-7 — In Table 19-2, add the following signals:

$\overline{\text{LSSD\_MODE}}$	Test	Test	Factory test. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper treatment.		I	19-10
L1_TSTCLK	Test	Test	Factory test. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper treatment.		I	19-10
L2_TSTCLK	Test	Test	Factory test. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper treatment.		I	19-10

19.2.2.3, 19-9 In Table 19-5, add the following signal descriptions:

$\overline{\text{LSSD\_MODE}}$	I	Used for factory test. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper treatment.
$\overline{\text{L1\_TSTCLK}}$	I	Used for factory test. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper treatment.
$\overline{\text{L2\_TSTCLK}}$	I	Used for factory test. Refer to the *MPC8560 Integrated Processor Hardware Specifications* for proper treatment.

19.3.1.2, 19-13 Change the last sentence in the first paragraph to, “Note that the transaction address is qualified with the bits described in Section 19.3.1.3, “Watchpoint Monitor Address Mask Register (WMAMR),” before being compared with WMAR. Note also that the contents of WMAR are not qualified with WMAMR.”

19.3.1.3, 19-13 Change the last part of the first sentence to be “...contains the mask that is applied to a transaction address before the address is compared with WMAR.”

19.3.2.2, 19-18 In the first sentence, replace “TBCR[AMD]” with “TBCR0[AMD]”. Also, replace the last sentence of the first paragraph with the following, “The transaction address is qualified with the bits described in Section 19.3.2.3, “Trace Buffer Address Mask Register (TBAMR), ” before being compared with TBAR. Note that the contents of TBAR are not qualified with TBAMR.”

19.3.2.3, 19-18 Change the last part of the first sentence to be “...contains the mask that is applied to a transaction address before the address is compared with TBAR.”

19.3.2.4, 19-19 In the last sentence of the first paragraph, replace “TBCR[TMD]” with “TBCR0[TMD]”.

19.4.6.1, 19-29 In Table 19-27, replace the “Function” description of the CMDBC field with the following:

“Byte count. Range: 32 to 1 where a value of 0 indicates 32 bytes.

00000 = 32 bytes

00001 = 1 byte

00010 = 2 bytes

...

11110 = 30 bytes

11111 = 31 bytes”

19.4.6.1, 19-30 and 19-31	Change all instances of “TRSEL” and “TRSEL1” to “IFSEL” and all instances of “TBCR” to “TBCR1”.
23.4.1, 23-7	In Table 23-2, change the name of bits 6–7 to read “MAD4–MAD3”. Also, replace the description with the following: Master address output pin x connection. Note that the address indexes are relative to FCC1; see Figure 23-7. These bits determine the number of ATM PHYs supported by FCC1 and FCC2. 00 FCC1 supports 7 PHYs; FCC2 supports 31 PHYs 01 FCC1 supports 15 PHYs; FCC2 supports 15 PHYs 10 Reserved 11 FCC1 supports 31 PHYs; FCC2 supports 7 PHYs
23.4.1, 23-10	In Figure 23-7, the pin names for FCC1 UTOPIA Rx addresses are incorrectly displayed as: RxAddr[1] = PC15 RxAddr[2] = PD16 RxAddr[3] = PD17 Correct the pin names to be: RxAddr[1] = PC12 RxAddr[2] = PC6 RxAddr[3] = PD29
25.1, 25-2	Replace the first three bullets with the following: • The maximum input clock is the internal CPM clock which is the core complex bus (CCB) clock divided by 3. • Maximum period of 2.6 seconds (at 100 MHz) • 10-ns resolution (at 100 MHz)
25.2, 25-2	Replace the first two bullets with: • Internal CPM clock (CCB/3) • Internal CPM clock divided by 16 (CCB/48) Also, replace the first four sentences of the following paragraph with: “The internal CPM clock is generated in the CPM clock synthesizer and defaults to the CCB clock frequency divided by 3. The user can either

choose that frequency or the frequency divided by 16 as the input to the prescaler of each timer.”

Also, replace the last two sentences of the next paragraph with, “ The best resolution of the timer is one clock cycle (10 ns at 100 MHz). The maximum period (when the reference value is all ones) is 268,435,456 cycles (2.6 seconds at 100 MHz).”

25.2.1, 25-3 — Replace the Note toward the bottom of the page with, “TGATEx is internally synchronized to the internal CPM clock. After the falling edge of TGATEx is recognized, the counter begins counting after one internal CPM clock cycle when working with the internal clock.”

35.11, 35-81 — Add the following clarifying note:

NOTE

Ensure that the transmit INTQs and the receive INTQs are programmed as separate queues.

39.4.6, 39-41 — Replace the last sentence in the first paragraph:
“These structures reside in a 1-Mbyte memory region defined by IMAEXTBASE, and may reside on either the 60x bus or the local bus, as defined by IMACNTL[DSB].”
with the following sentence:
“These structures reside in a 1-Mbyte memory region defined by IMAEXTBASE, and may reside on the local bus, as defined by IMACNTL[DSB].”

43.5, 43-12 — In Table 43-5, replace IMMR with CCSRBAR in footnote 1.

44.5, 44-10 — In Table 44-6, replace IMMR with CCSRBAR in footnote 1.

45.5, 45-17 — In Table 45-7, change SPICLK to $\overline{\text{SPISEL}}$ in the PC1 row.

Glossary of Terms and Abbreviations

The glossary contains an alphabetical list of terms, phrases, and abbreviations used in this reference manual.

A

Architecture. A detailed specification of requirements for a processor or computer system. It does not specify details of how the processor or computer system must be implemented; instead it provides a template for a family of compatible *implementations*.

Atomic access. A bus access that attempts to be part of a read-write operation to the same address uninterrupted by any other access to that address (the term refers to the fact that the transactions are indivisible). The PowerPC architecture implements atomic accesses through the **lwarx**/**stwcx.** instruction pair.

Autobaud. The process of determining a serial data rate by timing the width of a single bit.

B

Beat. A single state on the bus interface that may extend across multiple bus cycles. A transaction can be composed of multiple address or data *beats*.

Big endian. A byte-ordering method in memory where the address *n* of a word corresponds to the *most-significant byte*. In an addressed memory word, the bytes are ordered (left to right) 0, 1, 2, 3, with 0 being the *most-significant byte*. See *Little endian*.

Boundedly undefined. A characteristic of certain operation results that are not rigidly prescribed by the PowerPC architecture. Boundedly- undefined results for a given operation may vary among implementations and between execution attempts in the same implementation.

Although the architecture does not prescribe the exact behavior for when results are allowed to be boundedly undefined, the results of executing instructions in contexts where results are allowed to be boundedly undefined are constrained to ones that could have been achieved by executing an arbitrary sequence of defined instructions, in valid form, starting in the state the machine was in before attempting to execute the given instruction.

Breakpoint. A programmable event that forces the core to take a breakpoint exception.

Burst. A multiple-beat data transfer whose total size is typically equal to a cache block.

Bus clock. Clock that causes the bus state transitions.

Bus master. The owner of the address or data bus; the device that initiates or requests the transaction.

C

Cache. High-speed memory containing recently accessed data or instructions (subset of main memory).

Cache block. A small region of contiguous memory that is copied from memory into a *cache*. The size of a cache block may vary among processors; the maximum block size is one *page*. In PowerPC processors, *cache coherency* is maintained on a cache-block basis. Note that the term ‘cache block’ is often used interchangeably with ‘cache line.’

Cache coherency. An attribute wherein an accurate and common view of memory is provided to all devices that share the same memory system. Caches are coherent if a processor performing a read from its cache is supplied with data corresponding to the most recent value written to memory or to another processor’s cache.

Cache flush. An operation that removes from a cache any data from a specified address range. This operation ensures that any modified data within the specified address range is written back to main memory. This operation is generated typically by a Data Cache Block Flush (**dcbf**) instruction.

Caching-inhibited. A memory update policy in which the *cache* is bypassed and the load or store is performed to or from main memory.

Cast out. A *cache block* that must be written to memory when a cache miss causes a cache block to be replaced.

Changed bit. One of two *page history bits* found in each *page table entry* (PTE). The processor sets the changed bit if any store is performed into the *page*. See also *Page access history bits* and *Referenced*.

Clean. An operation that causes a cache block to be written to memory, if modified, and then left in a valid, unmodified state in the cache.

Clear. To cause a bit or bit field to register a value of zero. See also *Set*.

Completer. In PCI-X, a completer is the device addressed by a transaction (other than a split completion transaction). If a target terminates a transaction with

a split response, the completer becomes the initiator of the subsequent split completion.

Context synchronization. An operation that ensures that all instructions in execution complete past the point where they can produce an *exception*, that all instructions in execution complete in the context in which they began execution, and that all subsequent instructions are *fetched* and executed in the new context. Context synchronization may result from executing specific instructions (such as **isync** or **rfi**) or when certain events occur (such as an exception).

Copy-back operation. A cache operation in which a cache line is copied back to memory to enforce cache coherency. Copy-back operations consist of snoop push-out operations and cache cast-out operations.

D

Direct-mapped cache. A cache in which each main memory address can appear in only one location within the cache; operates more quickly when the memory request is a cache hit.

Double data rate. Memory that allows data transfers at the start and end of a clock cycle. thereby doubling the data rate.

E

Effective address (EA). The 32-bit address specified for a load, store, or an instruction fetch. This address is then submitted to the MMU for translation to either a *physical memory* or an I/O address.

Exclusive state. MEI state (E) in which only one caching device contains data that is also in system memory.

F

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.

Fetch. Retrieving instructions from either the cache or main memory and placing them into the instruction queue.

Flush. An operation that causes a cache block to be invalidated and the data, if modified, to be written to memory.

G

General-purpose register (GPR). Any of the 32 registers in the general-purpose register file. These registers provide the source operands and destination results for all integer data manipulation instructions. Integer load instructions move data from memory to GPRs and store instructions move data from GPRs to memory.

Guarded. The guarded attribute pertains to out-of-order execution. When a page is designated as guarded, instructions and data cannot be accessed out-of-order.

H

Harvard architecture. An architectural model featuring separate caches and other memory management resources for instructions and data.

I

IEEE 754. A standard written by the Institute of Electrical and Electronics Engineers that defines operations and representations of binary floating-point numbers.

Illegal instructions. A class of instructions that are not implemented for a particular PowerPC processor. These include instructions not defined by the PowerPC architecture. In addition, for 32-bit implementations, instructions that are defined only for 64-bit implementations are considered to be illegal instructions. For 64-bit implementations instructions that are defined only for 32-bit implementations are considered to be illegal instructions.

Implementation. A particular processor that conforms to the PowerPC architecture, but may differ from other architecture-compliant implementations for example in design, feature set, and implementation of *optional* features. The PowerPC architecture has many different implementations.

Imprecise exception. A type of *synchronous exception* that is allowed not to adhere to the precise exception model (see *Precise exceptions*). The PowerPC architecture allows only floating-point exceptions to be handled imprecisely.

Inbound ATMU windows. Mappings that perform address translation from the external address space to the local address space, attach attributes and transaction types to the transaction, and map the transaction to its target interface.

Inter-packet gap. The gap between the end of one Ethernet packet and the beginning of the next transmitted packet.

Integer unit. An execution unit in the core responsible for executing integer instructions.

In-order. An aspect of an operation that adheres to a sequential model. An operation is said to be performed in-order if, at the time that it is performed, it is known to be required by the sequential execution model.

Instruction latency. The total number of clock cycles necessary to execute an instruction and make ready the results of that instruction.

K

Kill. An operation that causes a *cache block* to be invalidated without writing any modified data to memory.

L

Latency. The number of clock cycles necessary to execute an instruction and make ready the results of that execution for a subsequent instruction.

L2 cache. Level-2 cache. See *Secondary cache*.

Least-significant bit (lsb). The bit of least value in an address, register, field, data element, or instruction encoding.

Least-significant byte (LSB). The byte of least value in an address, register, data element, or instruction encoding.

Little endian. A byte-ordering method in memory where the address *n* of a word corresponds to the *least-significant byte*. In an addressed memory word, the bytes are ordered (left to right) 3, 2, 1, 0, with 3 being the *most-significant byte*. See *Big endian*.

Local access window. Mapping used to translate a region of memory to a particular target interface, such as the DDR SDRAM controller or the PCI controller. The local memory map is defined by a set of eight local access windows. The size of each window can be configured from 4 Kbytes to 2 Gbytes.

M

Media access control (MAC) sublayer. Sublayer that provides a logical connection between the MAC and its peer station. Its primary responsibility is to initialize, control, and manage the connection with the peer station.

Medium-dependent interface (MDI) sublayer—Sublayer that defines different connector types for different physical media and PMD devices.

Media-independent interface (MII) sublayer. Sublayer that provides a standard interface between the MAC layer and the physical layer for 10/100-Mbps operations. It isolates the MAC layer and the physical layer, enabling the MAC layer to be used with various implementations of the physical layer.

MEI (modified/exclusive/invalid). *Cache coherency* protocol used to manage caches on different devices that share a memory system. Note that the PowerPC architecture does not specify the implementation of a MEI protocol to ensure cache coherency.

Memory access ordering. The specific order in which the processor performs load and store memory accesses and the order in which those accesses complete.

Memory-mapped accesses. Accesses whose addresses use the page or block address translation mechanisms provided by the MMU and that occur externally with the bus protocol defined for memory.

Memory coherency. An aspect of caching in which it is ensured that an accurate view of memory is provided to all devices that share system memory.

Memory consistency. Refers to agreement of levels of memory with respect to a single processor and system memory (for example, on-chip cache, secondary cache, and system memory).

Memory management unit (MMU). The functional unit that is capable of translating an effective (logical) *address* to a physical address, providing protection mechanisms, and defining caching methods.

Modified state. MEI state (M) in which one, and only one, caching device has the valid data for that address. The data at this address in external memory is not valid.

Most-significant bit (msb). The highest-order bit in an address, registers, data element, or instruction encoding.

Most-significant byte (MSB). The highest-order byte in an address, registers, data element, or instruction encoding.

N

NaN. An abbreviation for not a number; a symbolic entity encoded in floating-point format. There are two types of NaNs—signaling NaNs and quiet NaNs.

No-op. No-operation. A single-cycle operation that does not affect registers or generate bus activity.

O

OCeaN (On-chip network). Non-blocking crossbar switch fabric. Enables full duplex port connections at 128Gb/s concurrent throughput and independent per port transaction queuing and flow control. Permits high bandwidth, high performance, as well as the execution of multiple data transactions.

Outbound ATMU windows. Mappings that perform address translations from local 32-bit address space to the address spaces of 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.

P

Packet. A unit of binary data that can be routed through a network. Sometimes packet is used to refer to the frame plus the preamble and start frame delimiter (SFD).

Page. A region in memory. The OEA defines a page as a 4-Kbyte area of memory aligned on a 4-Kbyte boundary.

Page access history bits. The *changed* and *referenced* bits in the PTE keep track of the access history within the page. The referenced bit is set by the MMU whenever the page is accessed for a read or write operation. The changed bit is set when the page is stored into. See *Changed bit* and *Referenced bit*.

Page fault. A page fault is a condition that occurs when the processor attempts to access a memory location that does not reside within a *page* not currently resident in *physical memory*. On PowerPC processors, a page fault exception condition occurs when a matching, valid *page table entry* (PTE[V] = 1) cannot be located.

Page table. A table in memory is comprised of *page table entries*, or PTEs. It is further organized into eight PTEs per PTEG (page table entry group). The number of PTEGs in the page table depends on the size of the page table (as specified in the SDR1 register).

Page table entry (PTE). Data structures containing information used to translate *effective address* to physical address on a 4-Kbyte page basis. A PTE consists of 8 bytes of information in a 32-bit processor and 16 bytes of information in a 64-bit processor.

Physical coding sublayer (PCS). Sublayer responsible for encoding and decoding data stream to and from the MAC sublayer. Medium (1000BASEX) 8B/10B coding is used for fiber. Medium (1000BASET) 8B1Q coding is used for unshielded twisted pair (UTP).

Physical medium attachment (PMA) sublayer. Sublayer responsible for serializing code groups into a bit stream suitable for serial bit-oriented physical devices (SERDES) and vice versa. Synchronization is also performed for proper data decoding in this sublayer. The PMA sits between the PCS and the PMD sublayers.

Physical medium dependent (PMD) sublayer. Sublayer responsible for signal transmission. The typical PMD functionality includes amplifier, modulation, and wave shaping. Different PMD devices may support different media.

Physical memory. The actual memory that can be accessed through the system's memory bus.

Pipelining. A technique that breaks operations, such as instruction processing or bus transactions, into smaller distinct stages or tenures (respectively) so that a subsequent operation can begin before the previous one has completed.

Precise exceptions. A category of exception for which the pipeline can be stopped so instructions that preceded the faulting instruction can complete and subsequent instructions can be flushed and redispatched after exception handling has completed. See *Imprecise exceptions*.

Primary opcode. The most-significant 6 bits (bits 0–5) of the instruction encoding that identifies the type of instruction.

Program order. The order of instructions in an executing program. More specifically, this term is used to refer to the original order in which program instructions are fetched into the instruction queue from the cache.

Protection boundary. A boundary between *protection domains*.

Protection domain. A protection domain is a segment, a virtual page, a BAT area, or a range of unmapped effective addresses. It is defined only when the appropriate relocate bit in the MSR (IR or DR) is 1.

Q

Quad word. A group of 16 contiguous locations starting at an address divisible by 16.

Quiesce. To come to rest. The processor is said to quiesce when an exception is taken or a **sync** instruction is executed. The instruction stream is stopped at the decode stage and executing instructions are allowed to complete to create a controlled context for instructions that may be affected by out-of-order, parallel execution. See *Context synchronization*.

R

rA. The **r**A instruction field is used to specify a GPR to be used as a source or destination.

rB. The **r**B instruction field is used to specify a GPR to be used as a source.

rD. The **r**D instruction field is used to specify a GPR to be used as a destination.

rS. The **r**S instruction field is used to specify a GPR to be used as a source.

RapidIO. High-performance, packet-switched, interconnect architecture that provides reliability, increased bandwidth, and faster bus speeds in an intra-system interconnect. Designed to be compatible with integrated communications processors, host processors, and networking digital signal processors,

Record bit. Bit 31 (or the Rc bit) in the instruction encoding. When it is set, updates the condition register (CR) to reflect the result of the operation.

Reconciliation sublayer. Sublayer that maps the terminology and commands used in the MAC layer into electrical formats appropriate for the physical layer entities.

Referenced bit. One of two *page history bits* found in each *page table entry*. The processor sets the *referenced bit* whenever the page is accessed for a read or write. See also *Page access history bits*.

Requester. In PCI-X, a requester is an initiator that first introduces a transaction into the PCI-X domain. If a transaction is terminated with a split response, the requester becomes the target of the subsequent split completion.

Reservation. The processor establishes a reservation on a *cache block* of memory space when it executes an **lwarx** instruction to read a memory semaphore into a GPR.

Reservation station. A buffer between the dispatch and execute stages that allows instructions to be dispatched even though the results of instructions on which the dispatched instruction may depend are not available.

RISC (reduced instruction set computing). An *architecture* characterized by fixed-length instructions with nonoverlapping functionality and by a separate set of load and store instructions that perform memory accesses.

S

Secondary cache. A cache memory that is typically larger and has a longer access time than the primary cache. A secondary cache may be shared by multiple devices. Also referred to as L2, or level-2, cache.

Sequence. In PCI-X, a sequence is one or more transactions associated with carrying out a single logical transfer by a requester. Each transaction in the same sequence carries the same unique sequence ID.

Set (*v*)**.** To write a nonzero value to a bit or bit field; the opposite of *clear*. The term 'set' may also be used to generally describe the updating of a bit or bit field.

Set (*n*)**.** A subdivision of a *cache*. Cacheable data can be stored in a given location in one of the sets, typically corresponding to its lower-order address bits. Because several memory locations can map to the same location, cached data is typically placed in the set whose *cache block* corresponding to that address was used least recently. See *Set associative*.

Set associative. Aspect of cache organization in which the cache space is divided into sections, called *sets*. The cache controller associates a particular main memory address with the contents of a particular set, or region, within the cache.

Slave. The device addressed by a master device. The slave is identified in the address tenure and is responsible for supplying or latching the requested data for the master during the data tenure.

Snooping. Monitoring addresses driven by a bus master to detect the need for coherency actions.

Snoop push. Response to a snooped transaction that hits a modified cache block. The cache block is written to memory and made available to the snooping device.

Stall. An occurrence when an instruction cannot proceed to the next stage.

Sticky bit. A bit that when *set* must be cleared explicitly.

Superscalar machine. A machine that can issue multiple instructions concurrently from a conventional linear instruction stream.

Supervisor mode. The privileged operation state of a processor. In supervisor mode, software, typically the operating system, can access all control registers and can access the supervisor memory space, among other privileged operations.

Synchronization. A process to ensure that operations occur strictly *in order*. See *Context synchronization*.

Synchronous exception. An *exception* that is generated by the execution of a particular instruction or instruction sequence. There are two types of synchronous exceptions, *precise* and *imprecise*.

System memory. The physical memory available to a processor.

T

Time-division multiplex (TDM). A single serial channel used by several channels taking turns.

Tenure. The period of bus mastership. There can be separate address bus tenures and data bus tenures.

TLB (translation lookaside buffer). A cache that holds recently-used *page table entries*.

Throughput. The measure of the number of instructions that are processed per clock cycle.

Transaction. A complete exchange between two bus devices. A transaction is typically comprised of an address tenure and one or more data tenures, which may overlap or occur separately from the address tenure. A transaction may be minimally comprised of an address tenure only.

Transfer termination. Signal that refers to both signals that acknowledge the transfer of individual beats (of both single-beat transfer and individual beats of a burst transfer) and to signals that mark the end of the tenure.

U

User mode. The operating state of a processor used typically by application software. In user mode, software can access only certain control registers and can access only user memory space. No privileged operations can be performed. Also referred to as problem state.

V

Virtual address. An intermediate address used in the translation of an *effective address* to a physical address.

Virtual memory. The address space created using the memory management facilities of the processor. Program access to *virtual memory* is possible only when it coincides with *physical memory*.

W

Way. A location in the cache that holds a cache block, its tags, and status bits.

Word. A 32-bit data element.

Write-back. A cache memory update policy in which processor write cycles are directly written only to the cache. External memory is updated only indirectly, for example, when a modified cache block is *cast out* to make room for newer data.

Write-through. A cache memory update policy in which all processor write cycles are written to both the cache and memory.

Index 1
Register Index (Memory-Mapped Registers)

E

F

G

H

I

J

L

M

N

O

P

R

S

T

V

W

Index 2
General Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

W

X

Z

Index 3
CPM Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U