MPC8540 PowerQUICC III™ Integrated Host Processor Reference Manual

MPC8540RM
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

MPC8540RM
Rev. 1
07/2004

freescale™
semiconductor

Contents

Contents

Chapter 2
Memory Map

Contents

Contents

Part II
e500 Core Complex and L2 Cache

Chapter 5
Core Complex Overview

Contents

Chapter 6
Core Register Summary

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 10
Programmable Interrupt Controller

Contents

Contents

Contents

Contents

Chapter 13
Local Bus Controller

Contents

Contents

Contents

Chapter 14
Three-Speed Ethernet Controllers

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 16
PCI/PCI-X Bus Interface

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Chapter 19
Performance Monitor

Contents

Chapter 20
Debug Features and Watchpoint Facility

Contents

Chapter 21
10/100 Fast Ethernet Controller

Contents

Contents

Contents

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figures

Figure Number	Title	Page Number

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

Tables

About This Book

The primary objective of this reference manual is to define the functionality of the MPC8540. The MPC8540 integrates a PowerPC™ processor with system logic required for networking, storage, and general purpose embedded applications. 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 MPC8540 integrated host processor at an overview level. The following chapters are included:

- Chapter 1, "Overview," provides a high-level description of features and functionality of the MPC8540 integrated host processor. It describes the MPC8540, its interfaces, and its programming model. The functional operation of the MPC8540 with emphasis on peripheral functions is also described.
- Chapter 2, "Memory Map," describes the memory map of the MPC8540. An overview of the local address map is followed by a description of how local access windows are used to define the local address map. The inbound and outbound address translation mechanisms used to map to and from external memory spaces are described next. Finally, the configuration, control, and status registers are described, including a complete listing of all memory-mapped registers with cross references to 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 MPC8540.

Part II, "e500 Core Complex and L2 Cache," describes the many features of the MPC8540 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 MPC8540. 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 MPC8540 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 MPC8540.

  The ECM provides a mechanism for I/O-initiated transactions to snoop the core complex bus (CCB) of the e500 core in order to maintain coherency across cacheable local memory. It also provides a flexible, easily expandable switch-type structure for e500- and I/O-initiated transactions to be routed (dispatched) to target modules on the MPC8540.
- Chapter 9, "DDR Memory Controller," describes the DDR SDRAM memory controller of the MPC8540. This fully programmable controller supports most DDR memories available today, including both buffered and unbuffered devices. The built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. A large set of special features like DLL software override, crawl mode, and ECC error injection support rapid system debug.
- Chapter 10, "Programmable Interrupt Controller," describes the embedded programmable interrupt controller (PIC) of the MPC8540. This controller is an OpenPIC-compliant interrupt controller that provides interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them and delivering them to the CPU for servicing.
- Chapter 11, "I^2C Interface," describes the inter-IC (IIC or I^2C) bus controller of the MPC8540. This synchronous, serial, bidirectional, multi-master bus allows two-wire connection of devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters and LCDs. The MPC8540 powers up in boot sequencer mode which allows the I^2C controller to initialize configuration registers.
- Chapter 12, "DUART," describes the (dual) universal asynchronous receiver/transmitters (UARTs) which feature a PC16552D-compatible programming model. These independent UARTs are provided specifically to support system debugging.

- Chapter 13, “Local Bus Controller,” describes the local bus controller of the MPC8540. The main component of the local bus controller (LBC) is its memory controller which provides a seamless interface to many types of memory devices and peripherals. The memory controller is responsible for controlling eight memory banks shared by a high performance SDRAM machine, a general-purpose chip-select machine (GPCM), and up to three user-programmable machines (UPMs). As such, it supports a minimal glue logic interface to synchronous DRAM (SDRAM), SRAM, EPROM, flash EPROM, burstable RAM, regular DRAM devices, extended data output DRAM devices, and other peripherals.
- Chapter 14, “Three-Speed Ethernet Controllers,” describes the 2 three-speed Ethernet controllers on the MPC8540. These controllers provide 10/100/1000 Mbps Ethernet support with a complete set of media-independent interface options including GMII, RGMII, TBI, and RTBI. Each controller provides very high throughput using a captive DMA channel and direct connection to the MPC8540 memory coherency module.
- Chapter 15, “DMA Controller,” describes the four-channel general-purpose DMA controller of the MPC8540. The DMA controller transfers blocks of data, independent of the e500 core or external hosts. Data movement occurs among RapidIO and the local address space. The DMA controller has four high-speed channels. Both the e500 core and external masters can initiate a DMA transfer. All channels are capable of complex data movement and advanced transaction chaining.
- Chapter 16, “PCI/PCI-X Bus Interface,” describes the PCI/PCI-X controller of the MPC8540.
- Chapter 17, “RapidIO Interface,” describes the RapidIO controller of the MPC8540.

Part IV, “Global Functions and Debug,” defines other global blocks of the MPC8540. The following chapters are included:

- Chapter 18, “Global Utilities,” defines the global utilities of the MPC8540. These include power management, I/O device enabling, power-on-reset (POR) configuration monitoring, general-purpose I/O signal use, and multiplexing for the interrupt and local bus chip select signals
- Chapter 19, “Performance Monitor,” describes the performance monitor of the MPC8540. Note that the MPC8540 performance monitor is similar to but separate from the performance monitor implemented on the e500 core.
- Chapter 20, “Debug Features and Watchpoint Facility,” describes the debug features and watchpoint monitor of the MPC8540.
- Chapter 21, “10/100 Fast Ethernet Controller,” includes information about using the dual-speed Ethernet controller of the MPC8540 for debug purposes.
- Appendix A, “Revision History,” lists the major differences between revisions of the *MPC8540 PowerQUICC III Integrated Host 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*.
    - 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 memory-mapped registers.
- 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, $\overline{\textit{core int}}$
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.
R 0 W	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.
R FIELDNAME W	Indicates a read-only bit field in a memory-mapped register.
R W FIELDNAME	Indicates a write-only bit field in a memory-mapped register. Although these bits can be written to as ones or zeros, they are always read as zeros.

Signal Conventions

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

Table 1. Acronyms and Abbreviated Terms (continued)

Term	Meaning
CCSR	Configuration control and status register
CEPT	Conference des administrations Europeanes des Postes et Telecommunications (European Conference of Postal and Telecommunications Administrations).
COL	Collision
CRC	Cyclic redundancy check
CRS	Carrier sense
DDR	Double data rate
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory
DUART	Dual universal asynchronous receiver/transmitter
EA	Effective address
ECC	Error checking and correction
ECM	e500 coherency module
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

Table 1. Acronyms and Abbreviated Terms (continued)

Term	Meaning
JTAG	Joint Test Action Group
LAE	Local access error
LAW	Local access window
LBC	Local bus controller
LIFO	Last-in-first-out
LRU	Least recently used
LSB	Least-significant byte
lsb	Least-significant bit
LSU	Load/store unit
MAC	Multiply accumulate, media access control
MDI	Medium-dependent interface
MESI	Modified/exclusive/shared/invalid—cache coherency protocol
MII	Media independent interface
MMU	Memory management unit
MSB	Most-significant byte
msb	Most-significant bit
NMSI	Nonmultiplexed serial interface
No-op	No operation
OCeaN	On-chip network
OSI	Open systems interconnection
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

Table 1. Acronyms and Abbreviated Terms (continued)

Term	Meaning
RIO	Abbreviation occasionally used to refer to the RapidIO interface
RTOS	Real-time operating system
RWITM	Read with intent to modify
RWM	Read modify write
Rx	Receive
RxBD	Receive buffer descriptor
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 MPC8540 integrated host 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 MPC8540 integrated host processor. It describes the MPC8540, its interfaces, and its programming model. The functional operation of the MPC8540 with emphasis on peripheral functions is also described.

Chapter 2, “Memory Map,” describes the MPC8540 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 MPC8540.

Chapter 1
Overview

The MPC8540 integrates a PowerPC™ processor core with system logic required for networking, storage, and general-purpose embedded applications. The MPC8540 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 MPC8540 integrated microprocessor.

1.1 Introduction

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

In addition, the MPC8540 offers 256 Kbytes of L2 cache, 2 integrated 10/100/1Gb three-speed Ethernet controllers (TSECs), a 10/100 maintenance port, 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, a general-purpose I/O port, and a dual universal asynchronous receiver/transmitter (DUART). The high level of integration in the MPC8540 simplifies board design and offers significant bandwidth and performance.

1.2 MPC8540 Overview

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

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

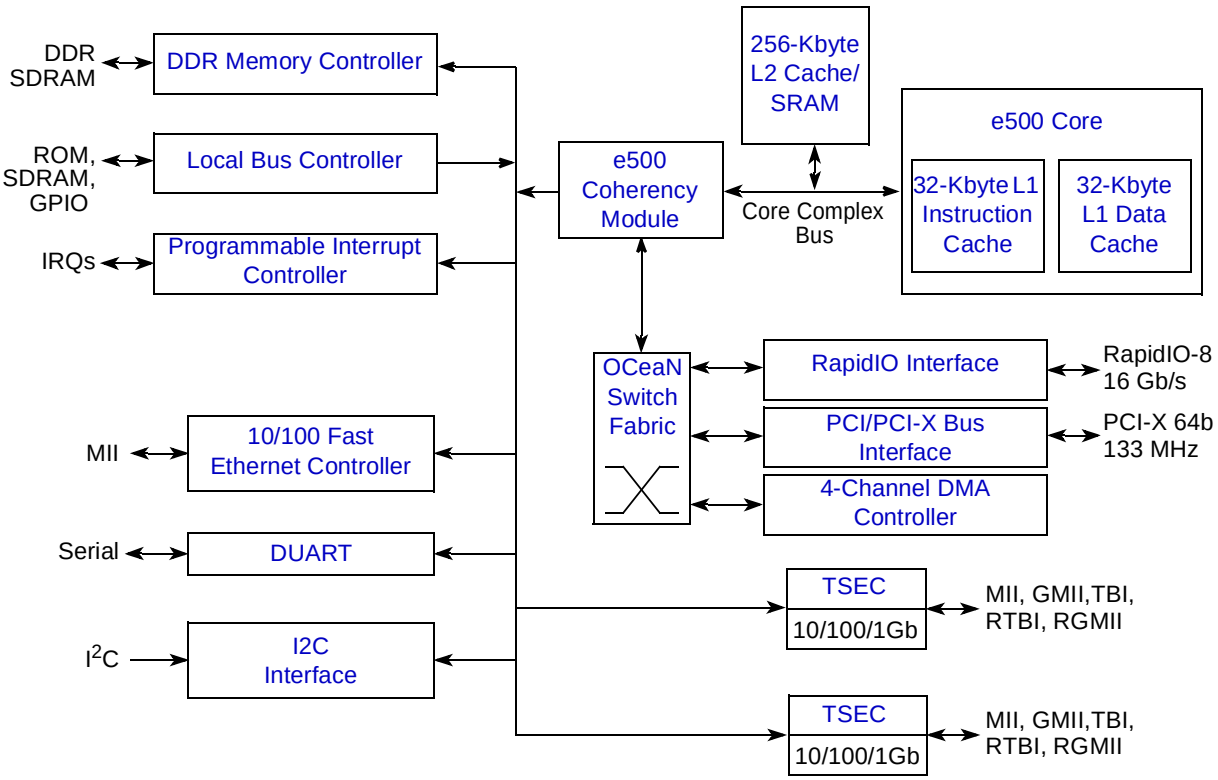


Figure 1-1. MPC8540 Block Diagram

1.2.1 Key Features

The following is an overview of the MPC8540 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 MPC8540 performance monitor described in Chapter 19, “Performance Monitor.”)

The e500 defines features that are not implemented on the MPC8540. It also generally defines some features that the MPC8540 implements more specifically. An understanding of these differences can be critical to ensure proper operations. These differences are summarized in Section 5.14, “MPC8540 Implementation Details,” in the *MPC8540 Integrated Processor Reference Manual*.

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

- 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
- DUART
  - Two 4-wire interfaces (SIN, SOUT, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$)
  - Programming model compatible with the original 16450 UART and the PC16550D
- 10/100 fast Ethernet controller (FEC)
  - Operates at 10 to 100 megabits per second (Mbps) as a device debug and maintenance port
- 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 MPC8540 Architecture Overview

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

1.3.1 e500 Core Overview

The MPC8540 uses the e500 microprocessor core complex. The e500 core has an internal PLL that allows independent optimization of the operating frequencies. The core frequencies are derived from either the primary PCI clock input or an external oscillator. For information regarding the e500 core refer to the following 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 MPC8540. It also generally defines some features that the MPC8540 implements more specifically. An understanding of these differences can be critical to ensure proper operation. These differences are summarized in Section 5.14, “MPC8540 Implementation Details,” in the *MPC8540 Integrated 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 On-Chip Memory Unit

The MPC8540 contains an internal 256-Kbyte memory array that can be configured as memory-mapped SRAM or as a look-aside L2 cache. The array can 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 MPC8540 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

1.3.2.1 On-Chip Memory as Memory-Mapped SRAM

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

1.3.2.2 On-Chip Memory as L2 Cache

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

The data array is organized as 1024 sets of 8 cache lines. Each cache line size is 32 bytes. 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.

1.3.3 e500 Coherency Module (ECM)

The e500 coherency module (ECM) provides a mechanism for I/O-initiated transactions to snoop the bus between the e500 core and the integrated L2 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.

1.3.4 DDR SDRAM Controller

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

The MPC8540 can be configured to retain the currently active SDRAM page for pipelined burst accesses. Page mode support of up to 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 MPC8540 supports ECC for system memory. Using ECC, the MPC8540 detects and corrects all single-bit errors and detects all double-bit errors and all errors within a nibble.

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

1.3.5 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 MPC8540 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 MPC8540.

1.3.6 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 MPC8540 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.

1.3.7 Boot Sequencer

The MPC8540 provides a boot sequencer that uses the I^2C interface to access an external serial ROM and loads the data into the MPC8540's configuration registers. The boot sequencer is enabled by a configuration pin sampled at the negation of the MPC8540 hardware reset signal. If enabled, the boot sequencer holds the MPC8540 processor core in reset until the boot sequence is complete. If the boot sequencer is not enabled, the processor core exits reset and fetches boot code in default configurations.

1.3.8 Dual Universal Asynchronous Receiver/Transmitter (DUART)

The MPC8540 includes a DUART intended for use in maintenance, bringing-up, and debugging of systems. The MPC8540 provides a standard four-wire handshake (SIN, SOUT, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$) for each port. The DUART is a slave interface. An interrupt is provided to the interrupt controller or optionally steered externally to allow device handshakes. Interrupts are generated for transmit, receive, line status, and MODEM status.

The MPC8540 DUART supports full-duplex operation. It is compatible with the PC16450 and PC16550 programming models. Also, 16-byte FIFOs are supported for both the transmitter and the receiver.

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

1.3.9 10/100 Fast Ethernet Controller

The MPC8540 provides a 10/100 fast Ethernet controller operating at 10 to 100 Mbps as a device debug and maintenance port. The gigabit three-speed Ethernet controllers (TSECs) are separate from this 10/100 fast Ethernet controller and are defined and described in Chapter 14, "Three-Speed Ethernet Controllers."

The 10/100 Ethernet standard supports the IEEE 802.3 Ethernet frame format, backward compatibility for installed media, and the use of full- or half-duplex CSMA/CD. The Ethernet protocol implements the bottom two layers of the open systems interconnection (OSI) 7-layer model, that is, the data-link and physical sublayers.

1.3.10 Local Bus Controller (LBC)

The MPC8540 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.

1.3.11 Three-Speed Ethernet Controllers (10/100/1Gb)

The MPC8540 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 MPC8540 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.

1.3.12 Integrated DMA

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

The 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).

1.3.13 PCI Controller

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

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

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

1.3.14 RapidIO Controller

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

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

The physical layer of the RapidIO unit can operate at up to 500 MHz. 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.

1.3.14.1 RapidIO Message Unit

The MPC8540's RapidIO messaging supports one inbox/outbox structure for data and one doorbell structure for messages. Both chaining and direct modes are provided for the outbox, and messages can hold up to 16 packets of 256 bytes, or a total of 4 Kbytes.

1.3.15 Power Management

In addition to low-voltage operation and dynamic power management in its execution units, the MPC8540 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 MPC8540 I/O interfaces as well. Only the interrupt controller and power management logic remain enabled so that the device can be awakened.

1.3.16 Clocking

The MPC8540 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.

1.3.17 Address Map

The MPC8540 supports a flexible physical address map. Conceptually, the address map consists of local space and external address space. The local address map is 4 Gbytes. The MPC8540 can be made part of a larger system address space through the mapping of translation windows. This functionality is included in the address translation and mapping units (ATMUs). Both inbound and outbound translation windows are provided. The ATMUs allows the MPC8540 to be part of larger address maps such as the PCI 64-bit address environment and the RapidIO environment.

1.3.18 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.

1.3.19 Processing Across the On-Chip Fabric

When processing across the on-chip fabric, the ATMUs at each fabric port are used to determine the flow of data across the MPC8540. The ATMUs at each fabric port are responsible for generating a fabric port destination ID as well as a new local device address. The port ID and local

address are based on the programmed destination of the transaction. The following is a general overview of how the ATMUs process transactions over the on-chip fabric. (Refer to Figure 1-2.)

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

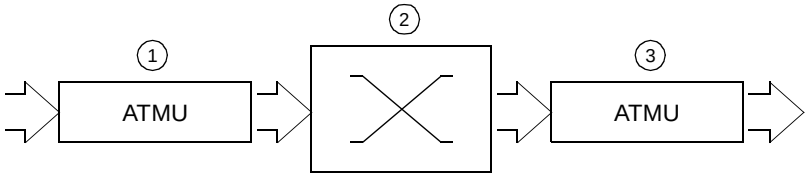


Figure 1-2. Processing Transactions Across the On-Chip Fabric

1.3.20 Data Processing with the e500 Coherency Module

Processing through the ECM is similar to processing across the on-chip fabric (in the sense of how data is received and transmitted) with the exception that the transaction passes through the ECM. The purpose of the ECM is to provide a means for any I/O transaction to maintain coherency with the cacheable DDR SDRAM and the local bus memory. However, simply 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 (across the on-chip fabric) are non-coherent.

1.4 MPC8540 Application Examples

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

1.4.1 MPC8540 Applications

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

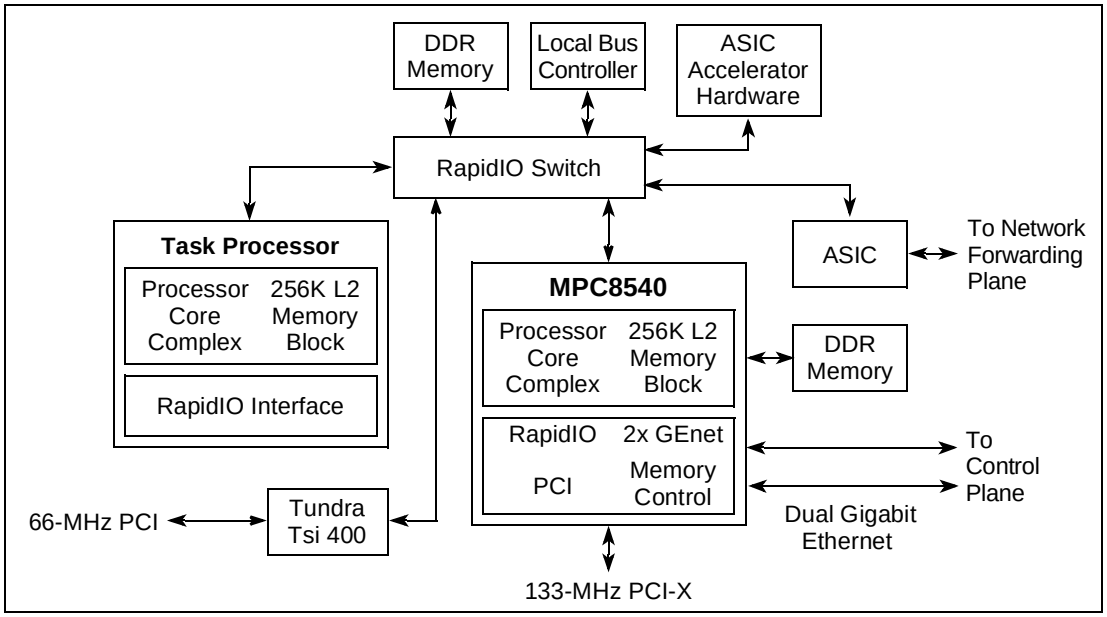


Figure 1-3. High-Performance Communication System Using SMP

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

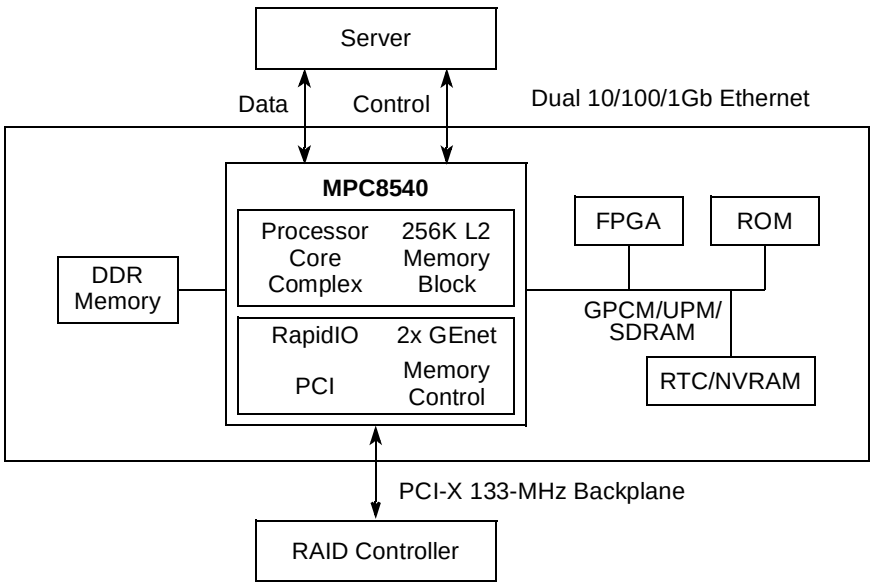


Figure 1-4. High-Performance Communication System Using MPC8540

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

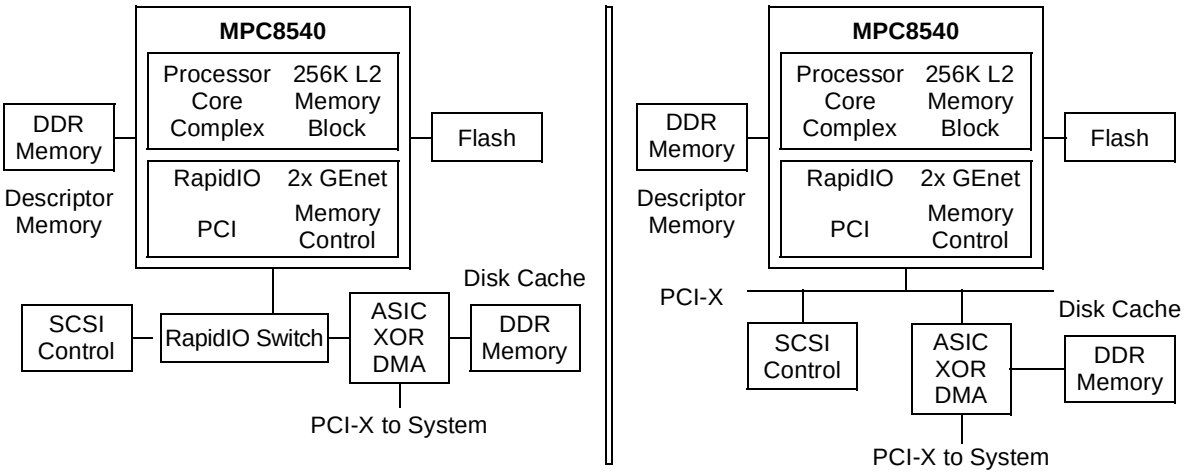


Figure 1-5. RAID Controller Application Using MPC8540

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

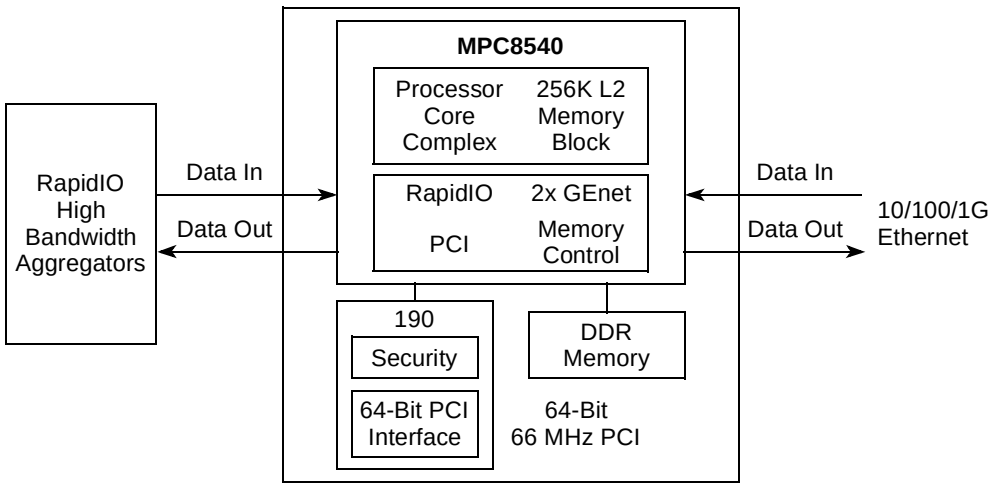


Figure 1-6. VPN Access Point Enabled by RapidIO and Ethernet

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

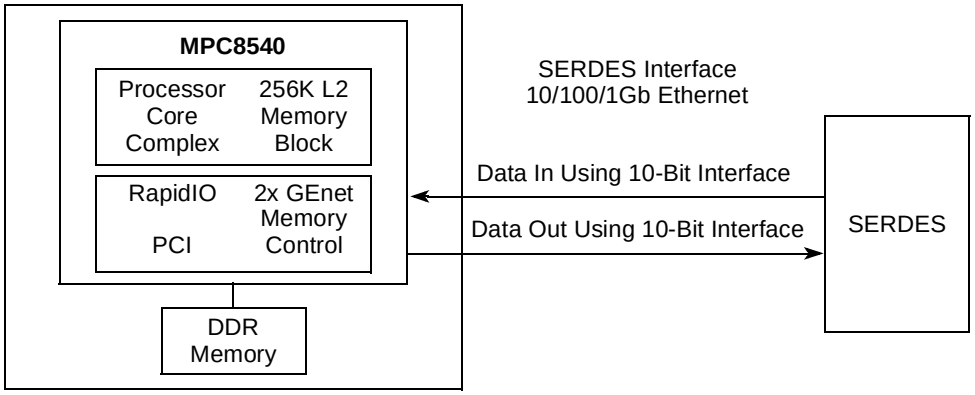


Figure 1-7. MPC8540 with SERDES

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

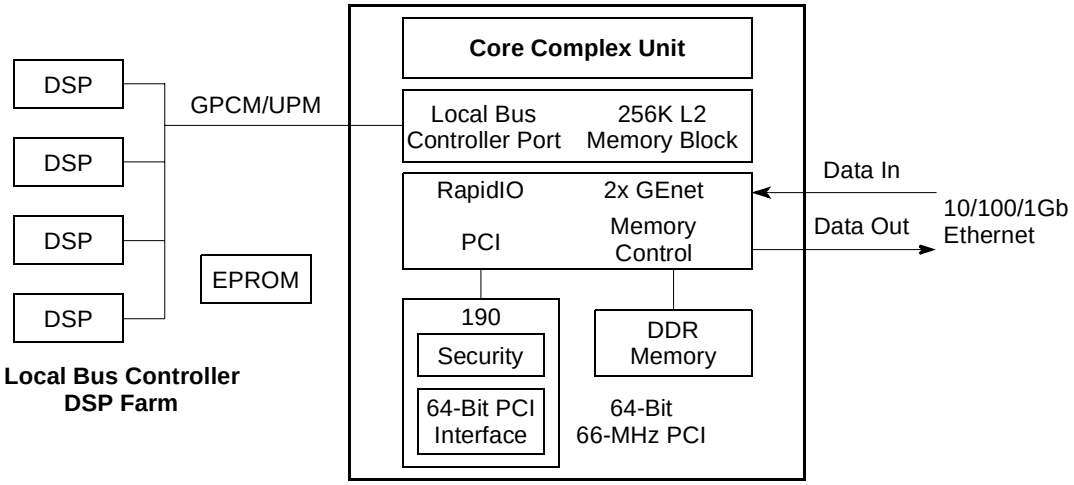


Figure 1-8. MPC8540 with a DSP Farm Enabled by General-Purpose Chip Select Machine or User Programmable Machine on the Local Bus Controller Port

Chapter 2
Memory Map

This chapter describes the MPC8540 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 MPC8540 provides an extremely flexible local memory map. The local memory map refers to the 32-bit address space seen by the processor as it accesses memory and I/O space. DMA engines also see this same local memory map. All memory accessed by the MPC8540 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.

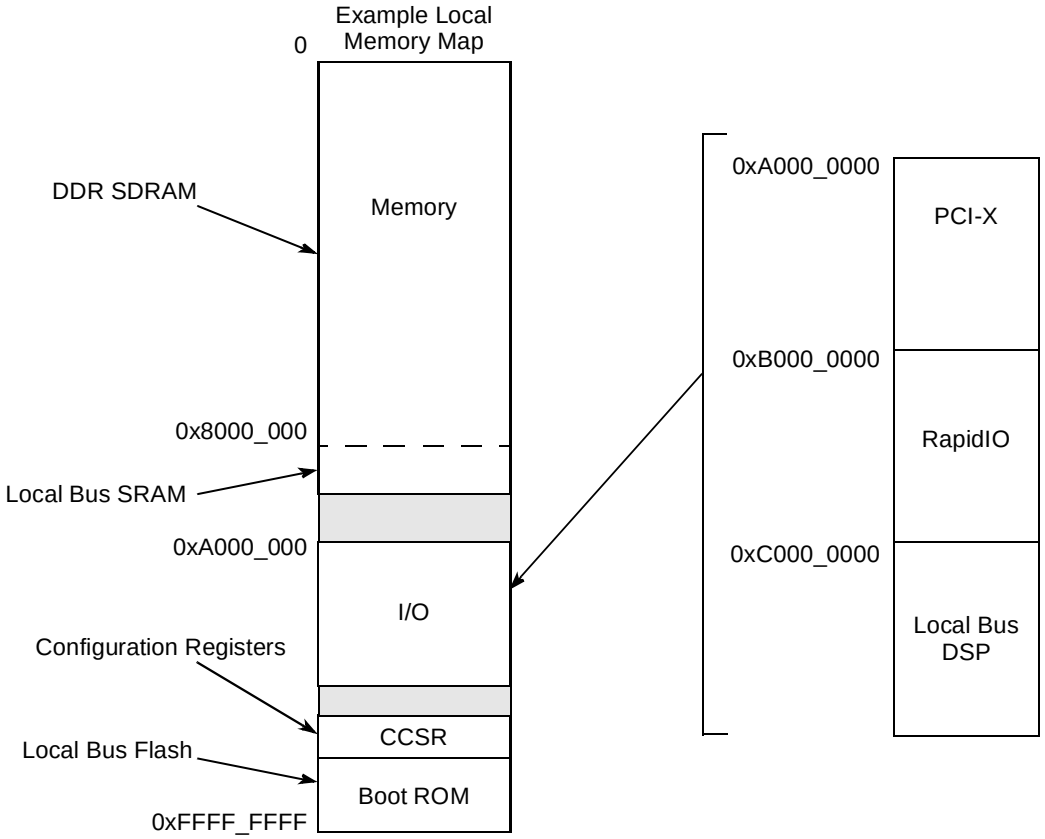


Figure 2-1. Local Memory Map Example

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 MPC8540. These are as follows:

- Mapping a local address to a target interface
- Assigning attributes to transactions
- Translating the local 32-bit address to an external address space
- Translating external addresses to the local 32-bit address space

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

Outbound ATMU windows perform the mapping from the local 32-bit address space to the address spaces of RapidIO or PCI/PCI-X, which may be much larger than the local space. Outbound ATMU windows also map attributes such as transaction type or priority level.

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

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

Table 2-3. Format of ATMU Window Definitions

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

Windows must be a power-of-two size. To perform a translation or mapping function, the address of the transaction is compared with the base address register of each window. The number of bits used in the comparison is dictated by each window’s size attribute. When an address hits a

window, if address translation is being performed, the new translated address is created by concatenating the window offset to the translation address. Again, the windows size attribute dictates how many bits are translated.

2.2.1 SRAM Windows

The on-chip memory array of the MPC8540 can be configured as a memory-mapped SRAM of 128 or 256 Kbytes. Configuration registers in the L2 cache controller set the base addresses and sizes for these windows. When enabled, these windows 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 MPC8540 to route a transaction from its source to the proper target. No address translation is performed. The base address defines the high order address bits that give the location of the window in the local address space. The window attributes enable the window, define its size, and specify the target interface.

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

The local access window registers exist as part of the local access block in the general utilities registers. See Section 2.3.3, “General Utilities Registers.” A detailed description of the local access window registers is given 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.

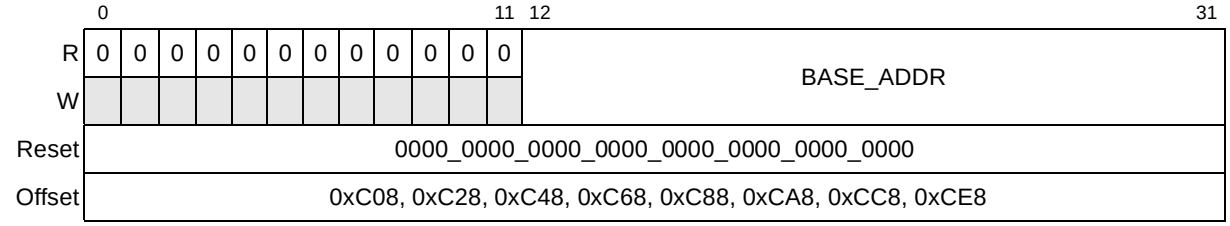


Figure 2-2. Local Access Window *n* Base Address Registers (LAWBAR0–LAWBAR7)

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.

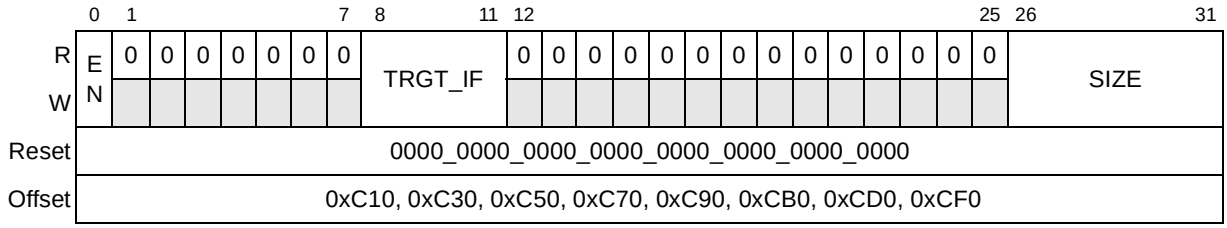


Figure 2-3. Local Access Window *n* Attributes Registers (LAWAR0–LAWAR7)

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 0001–0011Reserved 0100 Local bus memory controller 0101–1011Reserved 1100 RapidIO 1101–1110Reserved 1111 DDR SDRAM
12–25	—	Write reserved, read = 0
26–31	SIZE	Identifies the size of the window from the starting address. Window size is $2^{(SIZE+1)}$ bytes. 000000–001010Reserved 001011 4 Kbytes 001100 8 Kbytes 00110116 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 MPC8540 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 MPC8540, both the RapidIO and the PCI/PCI-X blocks have outbound address translation and mapping units (ATMUs).

The RapidIO controller has eight outbound ATMU windows plus a default window. If a transaction’s address does not hit any of the eight outbound ATMU windows, the translation actions defined by the default window are used. The default window is always enabled. See Section 17.3.2.2, “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 16.3.1.2, “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 MPC8540. It also refers to the mapping of transactions to a particular target interface and the assignment of transaction attributes. Both the RapidIO controller and the PCI/PCI-X controller have inbound address translation and mapping units (ATMUs).

2.2.5.1 RapidIO Inbound ATMU

The RapidIO controller has four inbound ATMU windows plus a default. If the inbound transaction’s address does not hit any of the four inbound ATMU windows, the translation actions defined by the default window are used. The default window is always enabled. See Section 17.3.2.2, “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 MPC8540 does not respond with an assertion of $\overline{\text{PCI\_DEVSEL}}$. See Section 16.3.1.3, “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 MPC8540 are contained within a 1-Mbyte address region. To allow for flexibility, the configuration, control, and status block is relocatable in the local address space. The local address map location of this register block is controlled by the configuration, control, and status registers base address register (CCSRBAR), see Section 4.3.1.1.2, "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.

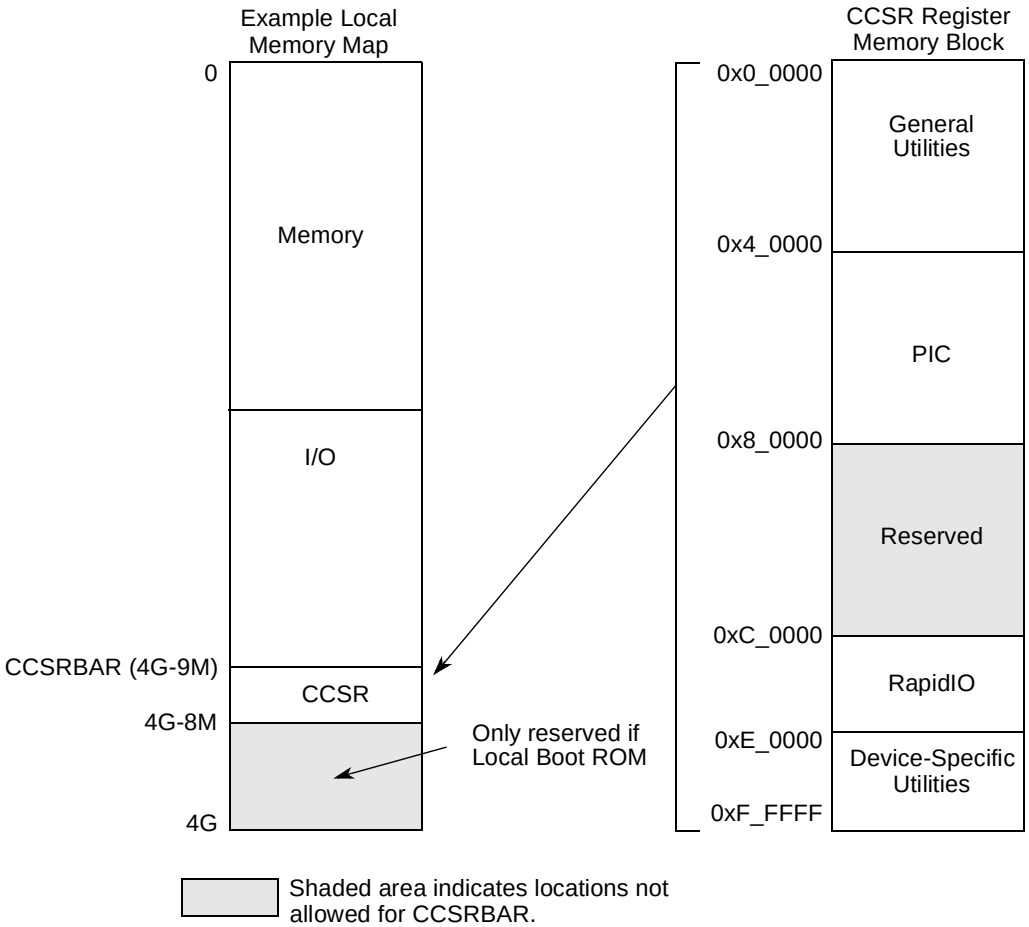


Figure 2-4. Top-Level Register Map Example

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 MPC8540.

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

The PCI base address for accessing the local CCSR memory is selectable through the PCI configuration and status register base address register (PCSRBAR), at offset 0x10, described in Section 16.3.2.11, "PCI Base Address Registers." An external PCI master sets this register by running a PCI configuration cycle to the MPC8540. Subsequent memory accesses by a PCI master to the PCI address range indicated by PCSRBAR are translated to the local address indicated by the current setting of CCSRBAR.

The RapidIO base address for accessing the local CCSR memory is selectable through the RapidIO LCSBA1CSR, defined in the RapidIO programming model, see Section 17.3.1.12, "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 MPC8540 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	Reserved
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.

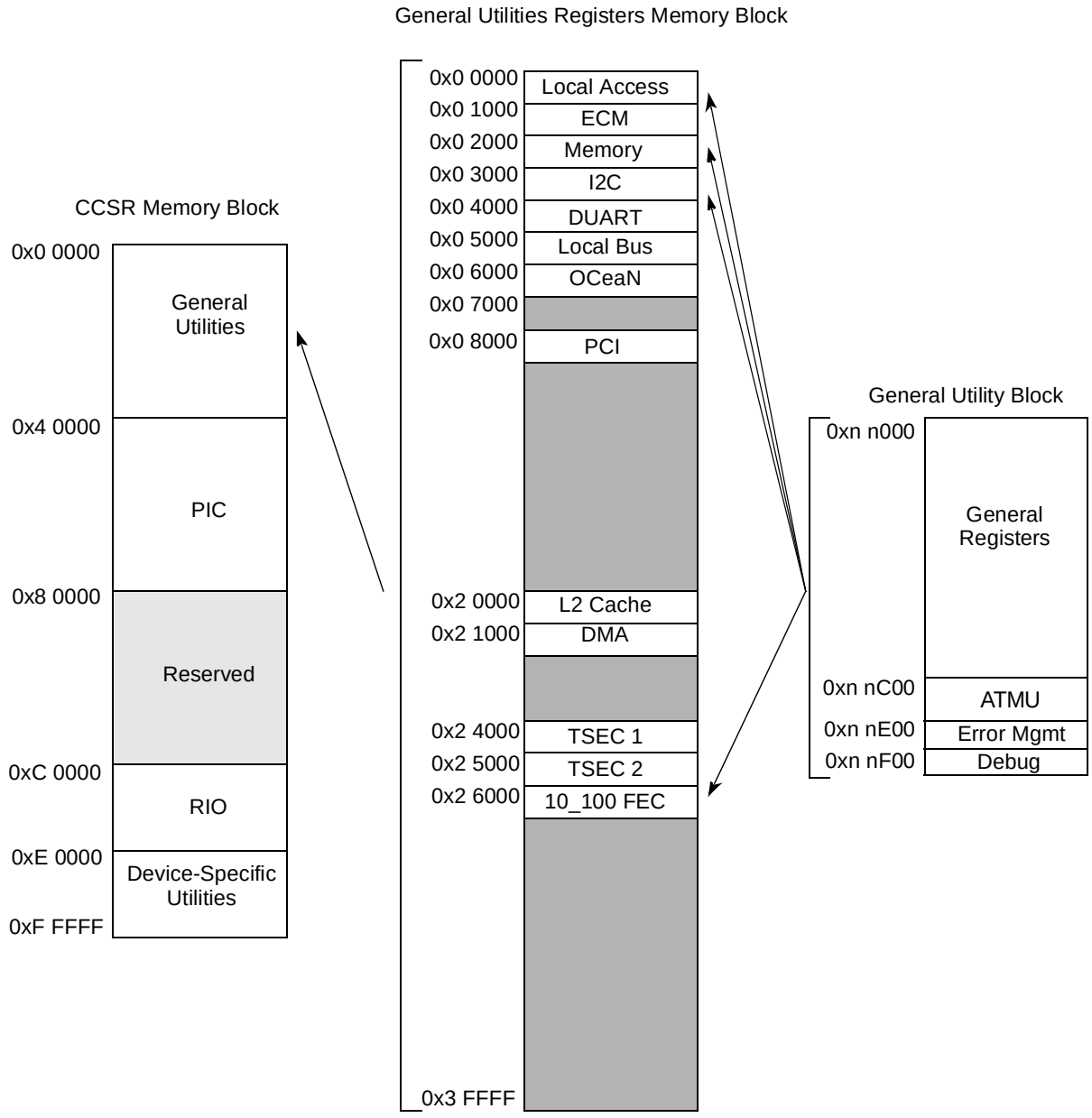


Figure 2-5. General Utilities Registers Mapping to Configuration, Control, and Status Memory Block

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 DUART and I^2C registers, which are accessed as bytes.

NOTE

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

2.3.4 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.

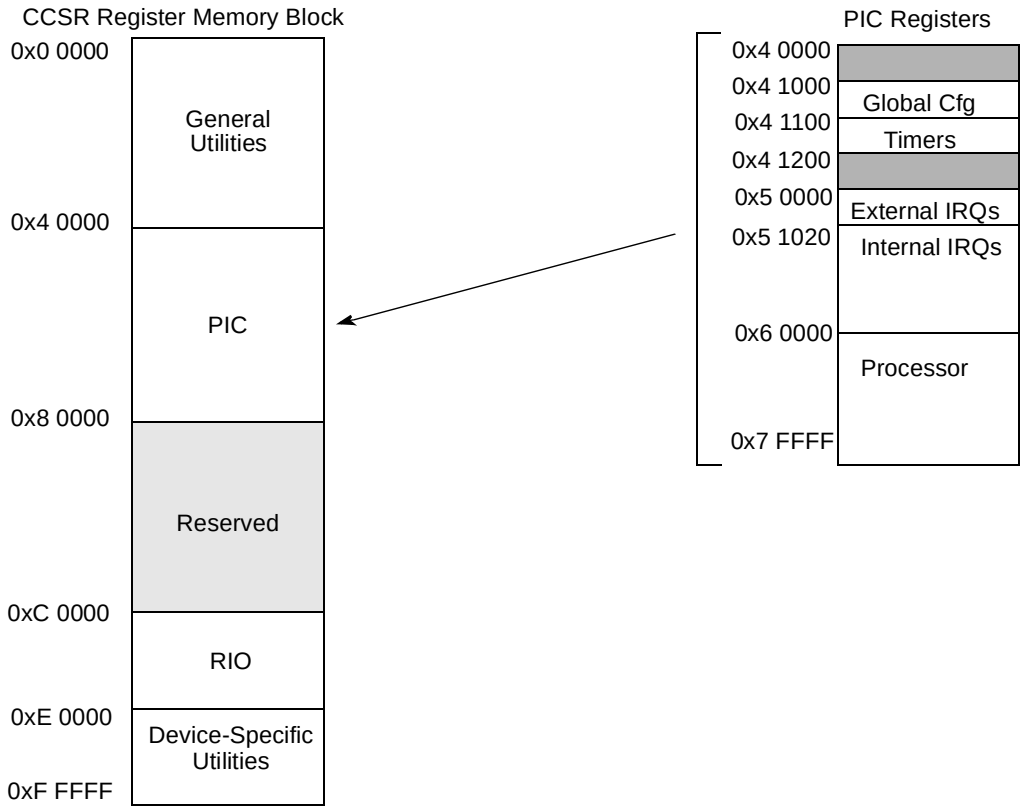


Figure 2-6. PIC Mapping to Configuration, Control, and Status Memory Block

2.3.5 RapidIO and CCSR

The RapidIO module uses 128 Kbytes of CCSR memory; refer to Figure 2-7. All registers are 32-bits wide and should only be accessed with 32-bit accesses. The 4-Kbyte RapidIO implementation block has the same internal organization as those defined for the general utilities described above.

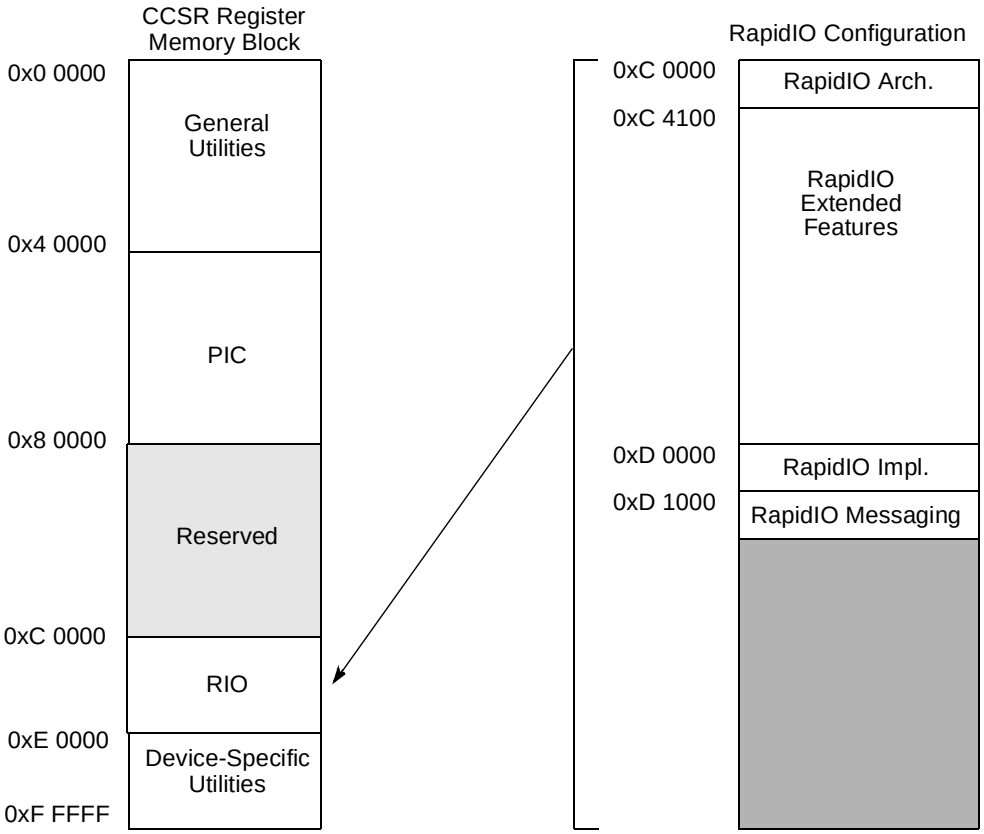


Figure 2-7. RapidIO Mapping to Configuration, Control, and Status Memory Block

2.3.6 Device-Specific Utilities

The device-specific registers consist of power management, performance monitors, and device-wide debug utilities (refer to Figure 2-8). 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.

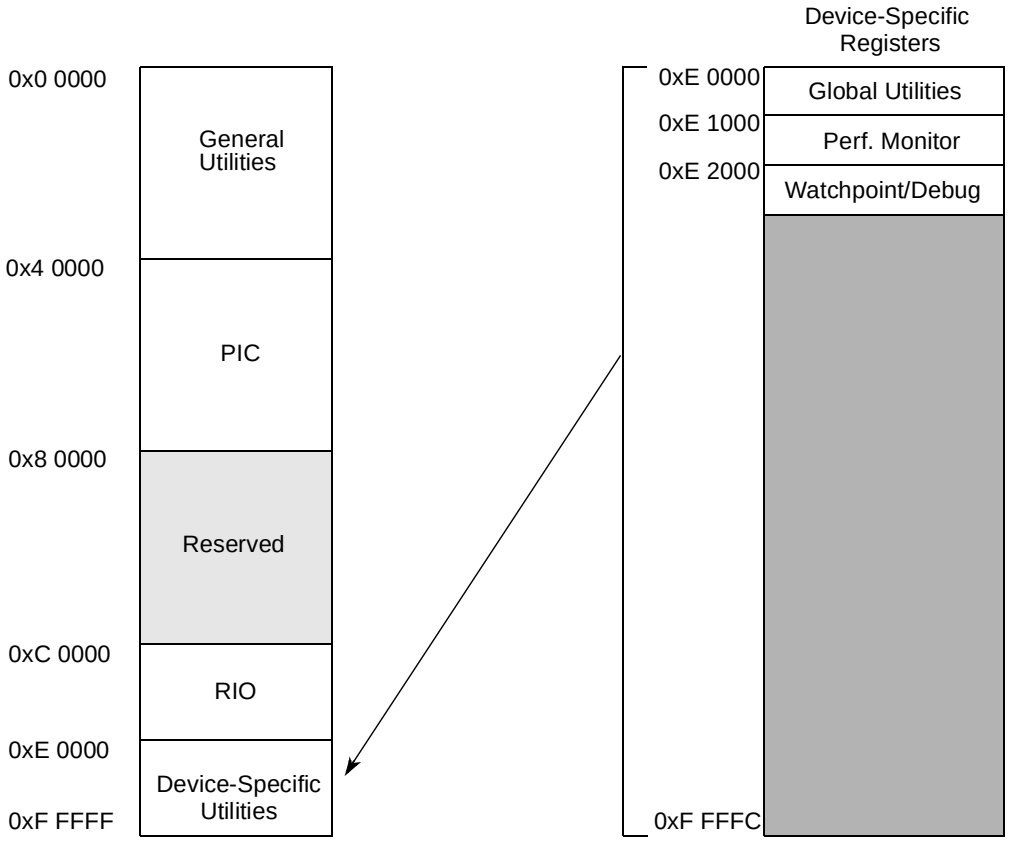


Figure 2-8. Device-Specific Register Mapping to Configuration, Control, and Status Memory Block

2.4 Complete CCSR Map

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

Table 2-9. Memory Map

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
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			

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
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
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

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_3010	I2CDR—I^{2}C data register	R/W	0x00	11.3.1.5/11-9
0x0_3014	I2CDFSRR—I^{2}C digital filter sampling rate register	R/W	0x10	11.3.1.6/11-10
DUART Registers				
0x0_4500	URBR—ULCR[DLAB] = 0 UART0 receiver buffer register	R	0x00	12.3.1.1/12-6
0x0_4500	UTHR—ULCR[DLAB] = 0 UART0 transmitter holding register	W	0x00	12.3.1.2/12-6
0x0_4500	UDLB—ULCR[DLAB] = 1 UART0 divisor least significant byte register	R/W	0x00	12.3.1.3/12-7
0x0_4501	UIER—ULCR[DLAB] = 0 UART0 interrupt enable register	R/W	0x00	12.3.1.4/12-9
0x0_4501	UDMB—ULCR[DLAB] = 1 UART0 divisor most significant byte register	R/W	0x00	12.3.1.3/12-7
0x0_4502	UIIR—ULCR[DLAB] = 0 UART0 interrupt ID register	R	0x01	12.3.1.5/12-10
0x0_4502	UFCR—ULCR[DLAB] = 0 UART0 FIFO control register	W	0x00	12.3.1.6/12-11
0x0_4502	UAFR—ULCR[DLAB] = 1 UART0 alternate function register	R/W	0x00	12.3.1.12/12-17
0x0_4503	ULCR—ULCR[DLAB] = x UART0 line control register	R/W	0x00	12.3.1.7/12-12
0x0_4504	UMCR—ULCR[DLAB] = x UART0 MODEM control register	R/W	0x00	12.3.1.8/12-14
0x0_4505	ULSR—ULCR[DLAB] = x UART0 line status register	R	0x60	12.3.1.9/12-15
0x0_4506	UMSR—ULCR[DLAB] = x UART0 MODEM status register	R	0x00	12.3.1.10/12-16
0x0_4507	USCR—ULCR[DLAB] = x UART0 scratch register	R/W	0x00	12.3.1.11/12-17
0x0_4510	UDSR—ULCR[DLAB] = x UART0 DMA status register	R	0x01	12.3.1.13/12-18
0x0_4600	URBR—ULCR[DLAB] = 0 UART1 receiver buffer register	R	0x00	12.3.1.1/12-6
0x0_4600	UTHR—ULCR[DLAB] = 0 UART1 transmitter holding register	W	0x00	12.3.1.2/12-6
0x0_4600	UDLB—ULCR[DLAB] = 1 UART1 divisor least significant byte register	R/W	0x00	12.3.1.3/12-7
0x0_4601	UIER—ULCR[DLAB] = 0 UART1 interrupt enable register	R/W	0x00	12.3.1.4/12-9
0x0_4601	UDMB_ULCR[DLAB] = 1 UART1 divisor most significant byte register	R/W	0x00	12.3.1.3/12-7
0x0_4602	UIIR—ULCR[DLAB] = 0 UART1 interrupt ID register	R	0x01	12.3.1.5/12-10
0x0_4602	UFCR—ULCR[DLAB] = 0 UART1 FIFO control register	W	0x00	12.3.1.6/12-11

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_4602	UAFR—ULCR[DLAB] = 1 UART1 alternate function register	R/W	0x00	12.3.1.12/12-17
0x0_4603	ULCR—ULCR[DLAB] = x UART1 line control register	R/W	0x00	12.3.1.7/12-12
0x0_4604	UMCR—ULCR[DLAB] = x UART1 MODEM control register	R/W	0x00	12.3.1.8/12-14
0x0_4605	ULSR—ULCR[DLAB] = x UART1 line status register	R	0x60	12.3.1.9/12-15
0x0_4606	UMSR—ULCR[DLAB] = x UART1 MODEM status register	R	0x00	12.3.1.10/12-16
0x0_4607	USCR—ULCR[DLAB] = x UART1 scratch register	R/W	0x00	12.3.1.11/12-17
0x0_4610	UDSR—ULCR[DLAB] = x UART1 DMA status register	R	0x01	12.3.1.13/12-18
Local Bus Controller Registers				
0x0_5000	BR0—Base register 0	R/W	0x0000_*nn*01[1]	13.3.1.1/13-11
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	13.3.1.2/13-12
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	13.3.1.3/13-18
0x0_5070	MAMR—UPMA mode register	R/W	0x0000_0000	13.3.1.4/13-19
0x0_5074	MBMR—UPMB mode register	R/W	0x0000_0000	13.3.1.4/13-19
0x0_5078	MCMR—UPMC mode register	R/W	0x0000_0000	13.3.1.4/13-19
0x0_5084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	13.3.1.5/13-21
0x0_5088	MDR—UPM data register	R/W	0x0000_0000	13.3.1.6/13-22
0x0_5094	LSDMR—SDRAM mode register	R/W	0x0000_0000	13.3.1.7/13-22
0x0_50A0	LURT—UPM refresh timer	R/W	0x0000_0000	13.3.1.8/13-24

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_50A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	13.3.1.9/13-25
0x0_50B0	LTESR—Transfer error status register	Read/ Bit-reset	0x0000_0000	13.3.1.10/13-26
0x0_50B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	13.3.1.11/13-27
0x0_50B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	13.3.1.12/13-28
0x0_50BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	13.3.1.13/13-29
0x0_50C0	LTEAR—Transfer error address register	R/W	0x0000_0000	13.3.1.14/13-30
0x0_50D0	LBCR—Configuration register	R/W	0x0000_0000	13.3.1.15/13-31
0x0_50D4	LCRR—Clock ratio register	R/W	0x8000_0008	13.3.1.16/13-32
PCI/X Registers				
PCI/X Configuration Access Registers				
0x0_8000	CFG_ADDR—PCI/X configuration address	R/W	0x0000_0000	16.3.1.1.1/16-18
0x0_8004	CFG_DATA—PCI/X configuration data	R/W	0x0000_0000	16.3.1.1.1/16-18
0x0_8008	INT_ACK—PCI/X interrupt acknowledge	R	0x0000_0000	16.3.1.1.3/16-20
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	16.3.1.2.1/16-21
0x0_8C04	POTEAR0—PCI/X outbound window 0 (default) translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-21
0x0_8C08	Reserved	—	—	
0x0_8C0C	Reserved	—	—	
0x0_8C10	POWAR0—PCI/X outbound window 0 (default) attributes register	R/W	0x8004_401F	16.3.1.2.4/16-22
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	16.3.1.2.1/16-21
0x0_8C24	POTEAR1—PCI/X outbound window 1 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-21
0x0_8C28	POWBAR1—PCI/X outbound window 1 base address register	R/W	0x0000_0000	16.3.1.2.3/16-22
0x0_8C2C	Reserved	—	—	
0x0_8C30	POWAR1—PCI/X outbound window 1 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
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	16.3.1.2.1/16-21
0x0_8C44	POTEAR2—PCI/X outbound window 2 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-21
0x0_8C48	POWBAR2—PCI/X outbound window 2 base address register	R/W	0x0000_0000	16.3.1.2.3/16-22
0x0_8C4C	Reserved	—	—	
0x0_8C50	POWAR2—PCI/X outbound window 2 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
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	16.3.1.2.1/16-21
0x0_8C64	POTEAR3—PCI/X outbound window 3 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-21
0x0_8C68	POWBAR3—PCI/X outbound window 3 base address register	R/W	0x0000_0000	16.3.1.2.3/16-22
0x0_8C6C	Reserved	—	—	
0x0_8C70	POWAR3—PCI/X outbound window 3 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
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	16.3.1.2.1/16-21
0x0_8C84	POTEAR4—PCI/X outbound window 4 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-21
0x0_8C88	POWBAR4—PCI/X outbound window 4 base address register	R/W	0x0000_0000	16.3.1.2.3/16-22
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	16.3.1.2.4/16-22
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	16.3.1.3.1/16-25
0x0_8DA4	Reserved	—	—	
0x0_8DA8	PIWBAR3—PCI/X inbound window 3 base address register	R/W	0x0000_0000	16.3.1.3.2/16-25
0x0_8DAC	PIWBEAR3—PCI/X inbound window 3 base extended address register	R/W	0x0000_0000	16.3.1.3.3/16-26
0x0_8DB0	PIWAR3—PCI/X inbound window 3 attributes register	R/W	0x0000_0000	16.3.1.3.4/16-26
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	16.3.1.3.1/16-25
0x0_8DC4	Reserved	—	—	
0x0_8DC8	PIWBAR2—PCI/X inbound window 2 base address register	R/W	0x0000_0000	16.3.1.3.2/16-25
0x0_8DCC	PIWBEAR2—PCI/X inbound window 2 base extended address register	R/W	0x0000_0000	16.3.1.3.3/16-26
0x0_8DD0	PIWAR2—PCI/X inbound window 2 attributes register	R/W	0x0000_0000	16.3.1.3.4/16-26
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	16.3.1.3.1/16-25
0x0_8DE4	Reserved	—	—	
0x0_8DE8	PIWBAR1—PCI/X inbound window 1 base address register	R/W	0x0000_0000	16.3.1.3.2/16-25
0x0_8DEC	Reserved	—	—	
0x0_8DF0	PIWAR1—PCI/X inbound window 1 attributes register	R/W	0x0000_0000	16.3.1.3.4/16-26
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	16.3.1.4.1/16-29
0x0_8E04	ERR_CAP_DR—PCI/X error capture disabled register	R/W	0x0000_0000	16.3.1.4.2/16-30
0x0_8E08	ERR_EN—PCI/X error enable register	R/W	0x0000_0000	16.3.1.4.3/16-31
0x0_8E0C	ERR_ATTRIB—PCI/X error attributes capture register	R/W	0x0000_0000	16.3.1.4.4/16-32
0x0_8E10	ERR_ADDR—PCI/X error address capture register	R/W	0x0000_0000	16.3.1.4.5/16-33
0x0_8E14	ERR_EXT_ADDR—PCI/X error extended address capture register	R/W	0x0000_0000	16.3.1.4.6/16-34
0x0_8E18	ERR_DL—PCI/X error data low capture register	R/W	0x0000_0000	16.3.1.4.7/16-34
0x0_8E1C	ERR_DH—PCI/X error data high capture register	R/W	0x0000_0000	16.3.1.4.8/16-34
0x0_8E20	GAS_TIMR—PCI/X gasket timer register	R/W	0x0000_0000	16.3.1.4.9/16-35
0x0_8E24	PCIX_TIMR—PCIX split completion timer register	R/W	0x0000_0000	16.3.1.4.10/16-36
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	7.3.1.3/7-10
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	7.3.1.3/7-10
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	7.3.1.3/7-10
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	7.3.1.3/7-10
0x2_0100	L2SRBAR0—L2 memory-mapped SRAM base address register 0	R/W	0x0000_0000	7.3.1.4/7-11
0x2_0108	L2SRBAR1—L2 memory-mapped SRAM base address register 1	R/W	0x0000_0000	7.3.1.4/7-11

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
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
DMA Registers				
General Registers				
0x2_1100	MR*n*—DMA 0 mode register	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1104	SR*n*—DMA 0 status register	Special	0x0000_0000	15.3.2.2/15-13
0x2_1108	Reserved	—	—	—
0x2_110C	CLNDAR*n*—DMA 0 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1110	SATR*n*—DMA 0 source attributes register	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1114	SAR*n*—DMA 0 source address register	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1118	DATR*n*—DMA 0 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-19
0x2_111C	DAR*n*—DMA 0 destination address register	R/W	0x0000_0000	15.3.2.7/15-21
0x2_1120	BCR*n*—DMA 0 byte count register	R/W	0x0000_0000	15.3.2.8/15-23
0x2_1124	Reserved	—	—	—
0x2_1128	NLNDAR*n*—DMA 0 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-23
0x2_1130	Reserved	—	—	—
0x2_1134	CLSDAR*n*—DMA 0 current list alternate base descriptor address register	R/W	0x0000_0000	15.3.2.10/15-24
0x2_1138	Reserved	—	—	—
0x2_113C	NLSDAR*n*—DMA 0 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-25
0x2_1140	SSR*n*—DMA 0 source stride register	R/W	0x0000_0000	15.3.2.12/15-25
0x2_1144	DSR*n*—DMA 0 destination stride register	R/W	0x0000_0000	15.3.2.13/15-26

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_1148–0x2_117C	Reserved	—	—	—
0x2_1180	MR*n*—DMA 1 mode register	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1184	SR*n*—DMA 1 status register	Special	0x0000_0000	15.3.2.2/15-13
0x2_1188	Reserved	—	—	—
0x2_118C	CLNDAR*n*—DMA 1 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1190	SATR*n*—DMA 1 source attributes register	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1194	SAR*n*—DMA 1 source address register	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1198	DATR*n*—DMA 1 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-19
0x2_119C	DAR*n*—DMA 1 destination address register	R/W	0x0000_0000	15.3.2.7/15-21
0x2_11A0	BCR*n*—DMA 1 byte count register	R/W	0x0000_0000	15.3.2.8/15-23
0x2_11A4	Reserved	—	—	—
0x2_11A8	NLNDAR*n*—DMA 1 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-23
0x2_11AC–0x2_11B0	Reserved	—	—	—
0x2_11B4	CLSDAR*n*—DMA 1 current list alternate base descriptor address register	R/W	0x0000_0000	15.3.2.10/15-24
0x2_11B8	Reserved	—	—	—
0x2_11BC	NLSDAR*n*—DMA 1 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-25
0x2_11C0	SSR*n*—DMA 1 source stride register	R/W	0x0000_0000	15.3.2.12/15-25
0x2_11C4	DSR*n*—DMA 1 destination stride register	R/W	0x0000_0000	15.3.2.13/15-26
0x2_11C8–0x2_11FC	Reserved	—	—	—
0x2_1200	MR*n*—DMA 2 mode register	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1204	SR*n*—DMA 2 status register	Special	0x0000_0000	15.3.2.2/15-13
0x2_1208	Reserved	—	—	—
0x2_120C	CLNDAR*n*—DMA 2 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1210	SATR*n*—DMA 2 source attributes register	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1214	SAR*n*—DMA 2 source address register	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1218	DATR*n*—DMA 2 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-19
0x2_121C	DAR*n*—DMA 2 destination address register	R/W	0x0000_0000	15.3.2.7/15-21
0x2_1220	BCR*n*—DMA 2 byte count register	R/W	0x0000_0000	15.3.2.8/15-23
0x2_1224	Reserved	—	—	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_1228	NLNDAR*n*—DMA 2 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-23
0x2_122C–0x2_1230	Reserved	—	—	—
0x2_1234	CLSDAR*n*—DMA 2 current list alternate base descriptor address register	R/W	0x0000_0000	15.3.2.10/15-24
0x2_1238	Reserved	—	—	—
0x2_123C	NLSDAR*n*—DMA 2 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-25
0x2_1240	SSR*n*—DMA 2 source stride register	R/W	0x0000_0000	15.3.2.12/15-25
0x2_1244	DSR*n*—DMA 2 destination stride register	R/W	0x0000_0000	15.3.2.13/15-26
0x2_1248–0x2_127C	Reserved	—	—	—
0x2_1280	MR*n*—DMA 3 mode register	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1284	SR*n*—DMA 3 status register	Special	0x0000_0000	15.3.2.2/15-13
0x2_1288	Reserved	—	—	—
0x2_128C	CLNDAR*n*—DMA 3 current link descriptor address register	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1290	SATR*n*—DMA 3 source attributes register	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1294	SAR*n*—DMA 3 source address register	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1298	DATR*n*—DMA 3 destination attributes register	R/W	0x0000_0000	15.3.2.6/15-19
0x2_129C	DAR*n*—DMA 3 destination address register	R/W	0x0000_0000	15.3.2.7/15-21
0x2_12A0	BCR*n*—DMA 3 byte count register	R/W	0x0000_0000	15.3.2.8/15-23
0x2_12A4	Reserved	—	—	—
0x2_12A8	NLNDAR*n*—DMA 3 next link descriptor address register	R/W	0x0000_0000	15.3.2.9/15-23
0x2_12AC–0x2_12B0	Reserved	—	—	—
0x2_12B4	CLSDAR*n*—DMA 3 current list alternate base descriptor address register	R/W	0x0000_0000	15.3.2.10/15-24
0x2_12B8	Reserved	—	—	—
0x2_12BC	NLSDAR*n*—DMA 3 next list descriptor address register	R/W	0x0000_0000	15.3.2.11/15-25
0x2_12C0	SSR*n*—DMA 3 source stride register	R/W	0x0000_0000	15.3.2.12/15-25
0x2_12C4	DSR*n*—DMA 3 destination stride register	R/W	0x0000_0000	15.3.2.13/15-26
0x2_12C8–0x2_12FC	Reserved	—	—	—
0x2_1300	DGSR—DMA general status register	Read	0x0000_0000	15.3.2.14/15-26
TSEC1 General Control and Status Registers				

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4000–0x2_400C	Reserved	R	0x0000_0000	—
0x2_4010	IEVENT—Interrupt event register	R/W	0x0000_0000	14.5.3.1.1/14-20
0x2_4014	IMASK—Interrupt mask register	R/W	0x0000_0000	14.5.3.1.2/14-23
0x2_4018	EDIS—Error disabled register	R/W	0x0000_0000	14.5.3.1.3/14-24
0x2_401C	Reserved	R	0x0000_0000	—
0x2_4020	ECNTRL—Ethernet control register	R/W	0x0000_0000	14.5.3.1.4/14-25
0x2_4024	MINFLR—Minimum frame length register	R/W	0x0000_0040	14.5.3.1.5/14-26
0x2_4028	PTV—Pause time value register	R/W	0x0000_0000	14.5.3.1.6/14-27
0x2_402C	DMACTRL—DMA control register	R/W	0x0000_0000	14.5.3.1.7/14-28
0x2_4030	TBIPA—TBI PHY address register	R/W	0x0000_0000	14.5.3.1.8/14-29
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	14.5.3.2.1/14-30
0x2_4050–0x2_4088	Reserved	R	0x0000_0000	—
0x2_408C	FIFO_TX_THR—FIFO transmit threshold register	R/W	0x0000_0100	14.5.3.2.2/14-31
0x2_4090–0x2_4094	Reserved	R	0x0000_0000	—
0x2_4098	FIFO_TX_STARVE—FIFO transmit starve register	R/W	0x0000_0080	14.5.3.2.3/14-32
0x2_409C	FIFO_TX_STARVE_SHUTOFF—FIFO transmit starve shutoff register	R/W	0x0000_0100	14.5.3.2.4/14-32
0x2_40A0–0x2_40FC	Reserved	R	0x0000_0000	—
TSEC1 Transmit Control and Status Registers				
0x2_4100	TCTRL—Transmit control register	R/W	0x0000_0000	14.5.3.3.1/14-33
0x2_4104	TSTAT—Transmit status register	R/W	0x0000_0000	14.5.3.3.2/14-34
0x2_4108	Reserved	R	0x0000_0000	—
0x2_410C	TBDLEN—TxBD data length register	R	0x0000_0000	14.5.3.3.3/14-35
0x2_4110	TXIC—Transmit interrupt coalescing configuration register	R/W	0x0000_0000	14.5.3.3.4/14-35
0x2_4114–0x2_4120	Reserved	R	0x0000_0000	—
0x2_4124	CTBPTR—Current TxBD pointer register	R	0x0000_0000	14.5.3.3.5/14-36
0x2_4128–0x2_4180	Reserved	R	0x0000_0000	—
0x2_4184	TBPTR—TxBD pointer register	R/W	0x0000_0000	14.5.3.3.6/14-37

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	14.5.3.3.7/14-37
0x2_4208–0x2_42AC	Reserved	R	0x0000_0000	—
0x2_42B0	OSTBD—Out-of-sequence TxBD register	R/W	0x0800_0000	14.5.3.3.8/14-38
0x2_42B4	OSTBDP—Out-of-sequence Tx data buffer pointer register	R/W	0x0000_0000	14.5.3.3.9/14-40
0x2_42B8–0x2_42FC	Reserved	R	0x0000_0000	—
TSEC1 Receive Control and Status Registers				
0x2_4300	RCTRL—Receive control register	R/W	0x0000_0000	14.5.3.4.1/14-41
0x2_4304	RSTAT—Receive status register	R/W	0x0000_0000	14.5.3.4.2/14-41
0x2_4308	Reserved	R	0x0000_0000	—
0x2_430C	RBDLEN—RxBD data length register	R	0x0000_0000	14.5.3.4.3/14-42
0x2_4310	RXIC—Receive interrupt coalescing configuration register	R/W	0x0000_0000	14.5.3.4.4/14-43
0x2_4314–0x2_4320	Reserved	R	0x0000_0000	—
0x2_4324	CRBPTR—Current RxBD pointer register	R	0x0000_0000	14.5.3.4.5/14-44
0x2_4328–0x2_433C	Reserved	R	0x0000_0000	—
0x2_4340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	14.5.3.4.6/14-44
0x2_4344–0x2_4380	Reserved	R	0x0000_0000	—
0x2_4384	RBPTR—RxBD pointer register	R/W	0x0000_0000	14.5.3.4.7/14-45
0x2_4388–0x2_4400	Reserved	R	0x0000_0000	—
0x2_4404	RBASE—RxBD base address register	R/W	0x0000_0000	14.5.3.4.8/14-45
0x2_4408–0x2_44FC	Reserved	R	0x0000_0000	—
TSEC1 MAC Registers				
0x2_4500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	14.5.3.6.1/14-49
0x2_4504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	14.5.3.6.2/14-51
0x2_4508	IPGIFG—Inter-packet gap/inter-frame gap register	R/W	0x4060_5060	14.5.3.6.3/14-52
0x2_450C	HAFDUP—Half-duplex register	R/W	0x00A1_F037	14.5.3.6.4/14-53
0x2_4510	MAXFRM—Maximum frame length register	R/W	0x0000_0600	14.5.3.6.5/14-54
0x2_4514–0x2_451C	Reserved	R	0x0000_0000	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4520	MIIMCFG—MII management configuration register	R/W	0x0000_0000	14.5.3.6.6/14-54
0x2_4524	MIIMCOM—MII Management command register	R/W	0x0000_0000	14.5.3.6.7/14-55
0x2_4528	MIIMADD—MII Management address register	R/W	0x0000_0000	14.5.3.6.8/14-56
0x2_452C	MIIMCON—MII Management control register	W	0x0000_0000	14.5.3.6.9/14-57
0x2_4530	MIIMSTAT—MII Management status register	R	0x0000_0000	14.5.3.6.10/14-57
0x2_4534	MIIMIND—MII Management indicator register	R	0x0000_0000	14.5.3.6.11/14-58
0x2_4538	Reserved	R	0x0000_0000	—
0x2_453C	IFSTAT—Interface status register	R/W	0x0000_0000	14.5.3.6.12/14-58
0x2_4540	MACSTNADDR1—Station address register, part 1	R/W	0x0000_0000	14.5.3.6.13/14-59
0x2_4544	MACSTNADDR2—Station address register, part 2	R/W	0x0000_0000	14.5.3.6.14/14-60
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	14.5.3.7.1/14-61
0x2_4684	TR127—Transmit and receive 65- to 127-byte frame counter register	R/W	0x0000_0000	14.5.3.7.2/14-61
0x2_4688	TR255—Transmit and receive 128- to 255-byte frame counter register	R/W	0x0000_0000	14.5.3.7.3/14-62
0x2_468C	TR511—Transmit and receive 256- to 511-byte frame counter register	R/W	0x0000_0000	14.5.3.7.4/14-62
0x2_4690	TR1K—Transmit and receive 512- to 1023-byte frame counter register	R/W	0x0000_0000	14.5.3.7.5/14-63
0x2_4694	TRMAX—Transmit and receive 1024- to 1518-byte frame counter register	R/W	0x0000_0000	14.5.3.7.6/14-63
0x2_4698	TRMGV—Transmit and receive 1519- to 1522-byte good VLAN frame count register	R/W	0x0000_0000	14.5.3.7.7/14-64
TSEC1 Receive Counters				
0x2_469C	RBYT—receive byte counter register	R/W	0x0000_0000	14.5.3.7.8/14-64
0x2_46A0	RPKT—receive packet counter register	R/W	0x0000_0000	14.5.3.7.9/14-65
0x2_46A4	RFCS—receive FCS error counter register	R/W	0x0000_0000	14.5.3.7.10/14-65
0x2_46A8	RMCA—receive multicast packet counter register	R/W	0x0000_0000	14.5.3.7.11/14-66
0x2_46AC	RBCA—receive broadcast packet counter register	R/W	0x0000_0000	14.5.3.7.12/14-66
0x2_46B0	RXCF—receive control frame packet counter register	R/W	0x0000_0000	14.5.3.7.13/14-67
0x2_46B4	RXPF—receive PAUSE frame packet counter register	R/W	0x0000_0000	14.5.3.7.14/14-67
0x2_46B8	RXUO—receive unknown OP code counter register	R/W	0x0000_0000	14.5.3.7.15/14-68

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_46BC	RALN—receive alignment error counter register	R/W	0x0000_0000	14.5.3.7.16/14-68
0x2_46C0	RFLR—receive frame length error counter register	R/W	0x0000_0000	14.5.3.7.17/14-69
0x2_46C4	RCDE—receive code error counter register	R/W	0x0000_0000	14.5.3.7.18/14-69
0x2_46C8	RCSE—receive carrier sense error counter register	R/W	0x0000_0000	14.5.3.7.19/14-70
0x2_46CC	RUND—receive undersize packet counter register	R/W	0x0000_0000	14.5.3.7.20/14-70
0x2_46D0	ROVR—receive oversize packet counter register	R/W	0x0000_0000	14.5.3.7.21/14-71
0x2_46D4	RFRG—receive fragments counter register	R/W	0x0000_0000	14.5.3.7.22/14-71
0x2_46D8	RJBR—receive jabber counter register	R/W	0x0000_0000	14.5.3.7.23/14-72
0x2_46DC	RDRP—receive drop register	R/W	0x0000_0000	14.5.3.7.24/14-72
TSEC1 Transmit Counters				
0x2_46E0	TBYT—Transmit byte counter register	R/W	0x0000_0000	14.5.3.7.25/14-73
0x2_46E4	TPKT—Transmit packet counter register	R/W	0x0000_0000	14.5.3.7.26/14-73
0x2_46E8	TMCA—Transmit multicast packet counter register	R/W	0x0000_0000	14.5.3.7.27/14-74
0x2_46EC	TBCA—Transmit broadcast packet counter register	R/W	0x0000_0000	14.5.3.7.28/14-74
0x2_46F0	TXPF—Transmit PAUSE control frame counter register	R/W	0x0000_0000	14.5.3.7.29/14-75
0x2_46F4	TDFR—Transmit deferral packet counter register	R/W	0x0000_0000	14.5.3.7.30/14-75
0x2_46F8	TEDF—Transmit excessive deferral packet counter register	R/W	0x0000_0000	14.5.3.7.31/14-76
0x2_46FC	TSCL—Transmit single collision packet counter register	R/W	0x0000_0000	14.5.3.7.32/14-76
0x2_4700	TMCL—Transmit multiple collision packet counter register	R/W	0x0000_0000	14.5.3.7.33/14-77
0x2_4704	TLCL—Transmit late collision packet counter register	R/W	0x0000_0000	14.5.3.7.34/14-77
0x2_4708	TXCL—Transmit excessive collision packet counter register	R/W	0x0000_0000	14.5.3.7.35/14-78
0x2_470C	TNCL—Transmit total collision counter register	R/W	0x0000_0000	14.5.3.7.36/14-78
0x2_4710	Reserved	R	0x0000_0000	—
0x2_4714	TDRP—Transmit drop frame counter register	R/W	0x0000_0000	14.5.3.7.37/14-79
0x2_4718	TJBR—Transmit jabber frame counter register	R/W	0x0000_0000	14.5.3.7.38/14-79
0x2_471C	TFCS—Transmit FCS error counter register	R/W	0x0000_0000	14.5.3.7.39/14-80
0x2_4720	TXCF—Transmit control frame counter register	R/W	0x0000_0000	14.5.3.7.40/14-80
0x2_4724	TOVR—Transmit oversize frame counter register	R/W	0x0000_0000	14.5.3.7.41/14-81
0x2_4728	TUND—Transmit undersize frame counter register	R/W	0x0000_0000	14.5.3.7.42/14-81
0x2_472C	TFRG—Transmit fragments frame counter register	R/W	0x0000_0000	14.5.3.7.43/14-82
TSEC1 General Registers				
0x2_4730	CAR1—Carry register one register	R/W	0x0000_0000	14.5.3.7.44/14-82

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_4734	CAR2—Carry register two register	R/W	0x0000_0000	14.5.3.7.45/14-84
0x2_4738	CAM1—Carry register one mask register	R/W	0xFE01_FFFF	14.5.3.7.46/14-85
0x2_473C	CAM2—Carry register two mask register	R/W	0x000F_FFFF	14.5.3.7.47/14-86
0x2_4740–0x2_47FC	Reserved	R	0x0000_0000	—
TSEC1 Hash Function Registers				
0x2_4800	IADDR0—Individual address register 0	R/W	0x0000_0000	14.5.3.8.1/14-88
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	14.5.3.8.2/14-88
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	—
TSEC1 Attribute Registers				
0x2_4BF8	ATTR—Attribute register	R/W	0x0000_0000	14.5.3.9.1/14-89
0x2_4BFC	ATTRELI—Attribute EL & EI register	R/W	0x0000_0000	14.5.3.9.2/14-90
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]			
FEC General Control and Status Registers				

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_6000–0x2_600C	Reserved	R	0x0000_0000	—
0x2_6010	IEVENT—Interrupt event register	R	0x0000_0000	21.5.3.1.1/21-13
0x2_6014	IMASK—Interrupt mask register	R/W	0x0000_0000	21.5.3.1.2/21-16
0x2_6018	EDIS—Error disabled register	R/W	0x0000_0000	21.5.3.1.3/21-18
0x2_601C–0x2_6020	Reserved	R	0x0000_0000	—
0x2_6024	MINFLR—Minimum frame length register	R/W	0x0000_0040	21.5.3.1.4/21-18
0x2_6028	PTV—Pause time value register	R/W	0x0000_0000	21.5.3.1.5/21-19
0x2_602C	DMACTRL—DMA control register	R/W	0x0000_0000	21.5.3.1.6/21-20
0x2_6030–0x2_6088	Reserved	R	0x0000_0000	—
FEC FIFO Control and Status Registers				
0x2_604C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	21.5.3.2.1/21-22
0x2_6050–0x2_6088	Reserved	R	0x0000_0000	—
0x2_608C	FIFO_TX_THR—FIFO transmit threshold register	R/W	0x0000_0080	21.5.3.2.2/21-23
0x2_6090–0x2_6094	Reserved	R	0x0000_0000	—
0x2_6098	FIFO_TX_STARVE—FIFO transmit starve register	R/W	0x0000_0020	21.5.3.3.4/21-27
0x2_609C	FIFO_TX_STARVE_SHUTOFF—FIFO transmit starve shutoff register	R/W	0x0000_0080	21.5.3.3.5/21-28
0x2_60A0–0x2_60FC	Reserved	R	0x0000_0000	—
FEC Transmit Control and Status Registers				
0x2_6100	TCTRL—Transmit control register	R/W	0x0000_0000	21.5.3.3.1/21-25
0x2_6104	TSTAT—Transmit status register	R/W	0x0000_0000	21.5.3.3.2/21-26
0x2_6108	Reserved	R	0x0000_0000	—
0x2_610C	TBDLEN—TxBD data length	R	0x0000_0000	21.5.3.3.3/21-27
0x2_6110–0x2_6120	Reserved	R	0x0000_0000	—
0x2_6124	CTBPTR—Current TxBD pointer register	R	0x0000_0000	21.5.3.3.4/21-27
0x2_6128–0x2_6180	Reserved	R	0x0000_0000	—
0x2_6184	TBPTR—TxBD pointer register	R/W	0x0000_0000	21.5.3.3.5/21-28
0x2_6188–0x2_6200	Reserved	R	0x0000_0000	—
0x2_6204	TBASE—TxBD base address register	R/W	0x0000_0000	21.5.3.3.6/21-28

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x2_6208–0x2_62AC	Reserved	R	0x0000_0000	—
0x2_62B0	OSTBD—Out-of-sequence TxBD register	R/W	0x0000_0000	21.5.3.3.7/21-29
0x2_62B4	OSTBDP—Out-of-sequence TxBD pointer register	R/W	0x0000_0000	21.5.3.3.8/21-31
0x2_62B8–0x2_62FC	Reserved	R	0x0000_0000	—
FEC Receive Control and Status Registers				
0x2_6300	RCTRL—Receive control register	R/W	0x0000_0000	21.5.3.4.1/21-31
0x2_6304	RSTAT—Receive status register	R/W	0x0000_0000	21.5.3.4.2/21-32
0x2_6308	Reserved	R	0x0000_0000	—
0x2_630C	RBDLEN—RxBD data length register	R	0x0000_0000	21.5.3.4.3/21-33
0x2_6310–0x2_6320	Reserved	R	0x0000_0000	—
0x2_6324	CRBPTR—Current RxBD pointer register	R	0x0000_0000	21.5.3.4.4/21-33
0x2_6328–0x2_633C	Reserved	R	0x0000_0000	—
0x2_6340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	21.5.3.4.5/21-34
0x2_6344–0x2_6380	Reserved	R	0x0000_0000	—
0x2_6384	RBPTR—RxBD pointer register	R/W	0x0000_0000	21.5.3.4.6/21-35
0x2_6388–0x2_6400	Reserved	R	0x0000_0000	—
0x2_6404	RBASE—RxBD base address register	R/W	0x0000_0000	21.5.3.4.7/21-35
0x2_6408–0x2_64FC	Reserved	R	0x0000_0000	—
FEC MAC Registers				
0x2_6500	MACCFG1—MAC configuration register #1	R/W	0x0000_0000	21.5.3.6.1/21-38
0x2_6504	MACCFG2—MAC configuration register #2	R/W	0x0000_7000	21.5.3.6.2/21-40
0x2_6508	IPGIFG—Inter-packet gap/inter-frame gap register	R/W	0x4060_5060	21.5.3.6.3/21-41
0x2_650C	HAFDUP—Half-duplex register	R/W	0x00A0_F037	21.5.3.6.4/21-41
0x2_6510	MAXFRM—Maximum frame length register	R/W	0x0000_0600	21.5.3.6.5/21-42
0x2_6514–0x2_6538	Reserved	R	0x0000_0000	—
0x2_653C	IFSTAT—Interface status register	R	0x0000_0000	21.5.3.6.6/21-43
0x2_6540	MACSTNADDR1—Station address register, part 1	R/W	0x0000_0000	21.5.3.6.7/21-43
0x2_6544	MACSTNADDR2—Station address register, part 2	R/W	0x0000_0000	21.5.3.6.8/21-44
0x2_6548–0x2_67FC	Reserved	R	0x0000_0000	—

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
FEC Hash Function Registers				
0x2_6800	IADDR0—Individual address register 0	R/W	0x0000_0000	21.5.3.7.1/21-45
0x2_6804	IADDR1—Individual address register 1	R/W	0x0000_0000	
0x2_6808	IADDR2—Individual address register 2	R/W	0x0000_0000	
0x2_680C	IADDR3—Individual address register 3	R/W	0x0000_0000	
0x2_6810	IADDR4—Individual address register 4	R/W	0x0000_0000	
0x2_6814	IADDR5—Individual address register 5	R/W	0x0000_0000	
0x2_6818	IADDR6—Individual address register 6	R/W	0x0000_0000	
0x2_681C	IADDR7—Individual address register 7	R/W	0x0000_0000	
0x2_6820–0x2_687C	Reserved	R	0x0000_0000	—
0x2_6880	GADDR0—Group address register 0	R/W	0x0000_0000	21.5.3.7.2/21-46
0x2_6884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x2_6888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x2_688C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x2_6890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x2_6894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x2_6898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x2_689C	GADDR7—Group address register 7	R/W	0x0000_0000	
0x2_68A0–0x2_6BF4	Reserved	R	0x0000_0000	—
FEC Attribute Registers				
0x2_6BF8	ATTR—Attribute register	R/W	0x0000_0000	21.5.3.8.1/21-46
0x2_6BFC	ATTRELI—Attribute EL & EI register	R/W	0x0000_0000	21.5.3.8.2/21-47
FEC Future Expansion Space				
0x2_6C00–0x2_6FFC	Reserved	R	0x0000_0000	—
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

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
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
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

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x4_1200–0x4_12F0	Reserved	—	—	—
0x4_1300	TCR—Timer control register	R/W	0x0000_0000	10.3.2.6/10-24
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	10.3.4.2/10-30
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	10.3.4.2/10-30
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	10.3.4.2/10-30
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	10.3.4.2/10-30
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	10.3.5.2/10-31
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

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
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
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

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
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
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

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
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
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

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
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

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
RapidIO Registers				
RapidIO Architectural Registers				
0xC_0000	DIDCAR—Device identity capability register	R	0x0002_0002	17.3.1.1/17-13
0xC_0004	DICAR—Device information capability register	R	0x8030_0020	17.3.1.2/17-14
0xC_0008	AIDCAR—Assembly identity capability register	R/W	0x0000_0000	17.3.1.3/17-14
0xC_000C	AICAR—Assembly information capability register	R/W	0x0000_0100	17.3.1.4/17-15
0xC_0010	PEFCAR—Processing element features capability register	R	0xE088_0009	17.3.1.5/17-15
0xC_0014	SPICAR—Switch port information capability register	R	0x0000_0100	17.3.1.6/17-17
0xC_0018	SOCAR—Source operations capability register	R	0x0600_FCF0	17.3.1.7/17-17
0xC_001C	DOCAR—Destination operations capability register	R	0x0600_FCF4	17.3.1.8/17-19
0xC_0040	MSR—Mailbox command and status register	R	0x0000_0000	17.3.1.9/17-22
0xC_0044	PWDCSR—Port-write and doorbell command and status register	R	0x0000_0020	17.3.1.10/17-22
0xC_004C	PELLCCSR—Processing element logical layer control command and status register	R	0x0000_0001	17.3.1.11/17-24
0xC_0058	Reserved	R	0x0000_0000	—
0xC_005C	LCSBA1CSR—Local configuration space base address 1 command and status register	R/W	0x0000_0000	17.3.1.12/17-24
0xC_0060	BDIDCSR—Base device ID command and status register	R/W	0x00[cfg]_0000	17.3.1.13/17-25
0xC_0068	HBDIDLCSR—Host base device ID lock command and status register	Special	0x0000_FFFF	17.3.1.14/17-26
0xC_006C	CTCSR—Component tag command and status register	R/W	0x0000_0000	17.3.1.15/17-26
0xC_0100	PMBH0CSR—8/16 LP-LVDS port maintenance block header 0 command and status register	R	0x0000_0002	17.3.1.16/17-27
0xC_0120	PLTOCCSR—Port link time-out control command and status register	R/W	0xFFFF_FF00	17.3.1.17/17-27
0xC_0124	PRTOCCSR—Port response time-out control command and status register	R/W	0xFFFF_FF00	17.3.1.18/17-28
0xC_013C	PGCCSR—Port general control command and status register	R/W	0x[cfg]000_0000	17.3.1.19/17-29
0xC_0140	PLMREQCSR—Port link maintenance request command and status register	R/W	0x0000_0000	17.3.1.20/17-29
0xC_0144	PLMRESPCSR—Port link maintenance response command and status register	R	0x0000_0000	17.3.1.21/17-30
0xC_0148	PLASCSR—Port local ackID status command and status register	R/W	0x0000_0000	17.3.1.22/17-31

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xC_0158	PESCSR—Port error and status command and status register	R/W	0x0000_0000	17.3.1.23/17-31
0xC_015C	PCCSR—Port control command and status register	R/W	0x4400_0000	17.3.1.24/17-33
Implementation Registers				
0xD_0000	CR—Configuration register	R/W	0x0000_0001	17.3.2.1.1/17-34
0xD_0010	PCR—Port configuration register	R/W	0x0000_0010	17.3.2.1.2/17-35
0xD_0014	PEIR—Port error injection register	R/W	0x0000_0000	17.3.2.1.3/17-35
ATMU Registers				
0xD_0C00	ROWTAR0—RapidIO outbound window translation address register 0	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0C10	ROWAR0—RapidIO outbound window attributes register 0	R/W	0x8004_401F	17.3.2.2.3/17-39
0xD_0C20	ROWTAR1—RapidIO outbound window translation address register 1	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0C28	ROWBAR1—RapidIO outbound window base address register 1	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0C30	ROWAR1—RapidIO outbound window attributes register 1	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0C40	ROWTAR2—RapidIO outbound window translation address register 2	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0C48	ROWBAR2—RapidIO outbound window base address register 2	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0C50	ROWAR2—RapidIO outbound window attributes register 2	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0C60	ROWTAR3—RapidIO outbound window translation address register 3	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0C68	ROWBAR3—RapidIO outbound window base address register 3	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0C70	ROWAR3—RapidIO outbound window attributes register 3	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0C80	ROWTAR4—RapidIO outbound window translation address register 4	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0C88	ROWBAR4—RapidIO outbound window base address register 4	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0C90	ROWAR4—RapidIO outbound window attributes register 4	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0CA0	ROWTAR5—RapidIO outbound window translation address register 5	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0CA8	ROWBAR5—RapidIO outbound window base address register 5	R/W	0x0000_0000	17.3.2.2.2/17-38

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_0CB0	ROWAR5—RapidIO outbound window attributes register 5	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0CC0	ROWTAR6—RapidIO outbound window translation address register 6	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0CC8	ROWBAR6—RapidIO outbound window base address register 6	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0CD0	ROWAR6—RapidIO outbound window attributes register 6	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0CE0	ROWTAR7—RapidIO outbound window translation address register 7	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0CE8	ROWBAR7—RapidIO outbound window base address register 7	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0CF0	ROWAR7—RapidIO outbound window attributes register 7	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0D00	ROWTAR8—RapidIO outbound window translation address register 8	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0D08	ROWBAR8—RapidIO outbound window base address register 8	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0D10	ROWAR8—RapidIO outbound window attributes register 8	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0D60	RIWTAR4—RapidIO inbound window translation address register 4	R/W	0x0000_0000	17.3.2.2.4/17-40
0xD_0D68	RIWBAR4—RapidIO inbound window base address register 4	R/W	0x0000_0000	17.3.2.2.5/17-41
0xD_0D70	RIWAR4—RapidIO inbound window attributes register 4	R/W	0x0004_401F	17.3.2.2.6/17-42
0xD_0D80	RIWTAR3—RapidIO inbound window translation address register 3	R/W	0x0000_0000	17.3.2.2.4/17-40
0xD_0D88	RIWBAR3—RapidIO inbound window base address register 3	R/W	0x0000_0000	17.3.2.2.5/17-41
0xD_0D90	RIWAR3—RapidIO inbound window attributes register 3	R/W	0x0004_401F	17.3.2.2.6/17-42
0xD_0DA0	RIWTAR2—RapidIO inbound window translation address register 2	R/W	0x0000_0000	17.3.2.2.4/17-40
0xD_0DA8	RIWBAR2—RapidIO inbound window base address register 2	R/W	0x0000_0000	17.3.2.2.5/17-41
0xD_0DB0	RIWAR2—RapidIO inbound window attributes register 2	R/W	0x0004_401F	17.3.2.2.6/17-42
0xD_0DC0	RIWTAR1—RapidIO inbound window translation address register 1	R/W	0x0000_0000	17.3.2.2.4/17-40
0xD_0DC8	RIWBAR1—RapidIO inbound window base address register 1	R/W	0x0000_0000	17.3.2.2.5/17-41

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_0DD0	RIWAR1—RapidIO inbound window attributes register 1	R/W	0x0004_401F	17.3.2.2.6/17-42
0xD_0DE0	RIWTAR0—RapidIO inbound window translation address register 0	R/W	0x0000_0000	17.3.2.2.4/17-40
0xD_0DF0	RIWAR0—RapidIO inbound window attributes register 0	R/W	0x8004_401F	17.3.2.2.6/17-42
Error Management Registers				
0xD_0E00	PNFEDR—Port notification/fatal error detect register	R/W	0x0000_0000	17.3.2.3.1/17-44
0xD_0E04	PNFEDiR—Port notification/fatal error detect disable register	R/W	0x0000_0000	17.3.2.3.2/17-47
0xD_0E08	PNFEIER—Port notification/fatal error interrupt enable register	R/W	0x0000_0000	17.3.2.3.3/17-49
0xD_0E0C	PECSR—Port error capture status register	R/W	0x0000_0000	17.3.2.3.4/17-52
0xD_0E10	EPCR0—Error packet capture register 0	R/W	0x0000_0000	17.3.2.3.5/17-52
0xD_0E14	EPCR1—Error packet capture register 1	R/W	0x0000_0000	17.3.2.3.6/17-53
0xD_0E18	EPCR2—Error packet capture register 2	R/W	0x0000_0000	17.3.2.3.16/17-58
0xD_0E20	PREDR—Port recoverable error detect register	R/W	0x0000_0000	17.3.2.3.23/17-61
0xD_0E28	PERTR—Port error recovery threshold register	R/W	0x00FF_0000	17.3.2.3.24/17-64
0xD_0E2C	PRTR—Port retry threshold register	R/W	0x00FF_0000	17.3.2.3.25/17-64
RapidIO Message Unit				
RapidIO Outbound Message Registers				
0xD_1000	OMR—Outbound mode register	R/W	0x0000_0000	17.3.3.1.1/17-65
0xD_1004	OSR—Outbound status register	R/W	0x0000_0000	17.3.3.1.2/17-67
0xD_100C	ODQDPAR—Outbound descriptor queue dequeue pointer address register	R/W	0x0000_0000	17.3.3.1.3/17-68
0xD_1014	OSAR—Outbound source address register	R/W	0x0000_0000	17.3.3.1.4/17-69
0xD_1018	ODPR—Outbound destination port register	R/W	0x0000_0000	17.3.3.1.5/17-70
0xD_101C	ODATR—Outbound destination attributes register	R/W	0x0006_0000	17.3.3.1.6/17-70
0xD_1020	ODCR—Outbound double-word count register	R/W	0x0000_0000	17.3.3.1.7/17-71
0xD_1028	ODQEPAR—Outbound descriptor queue enqueue pointer address register	R/W	0x0000_0000	17.3.3.1.8/17-72
RapidIO Inbound Message Registers				
0xD_1060	IMR—Inbound mailbox mode register	R/W	0x0000_0000	17.3.3.2.1/17-72
0xD_1064	ISR—Inbound mailbox status register	R/W	0x0000_0000	17.3.3.2.2/17-74
0xD_106C	IFQDPAR—Inbound frame queue dequeue pointer address register	R/W	0x0000_0000	17.3.3.2.3/17-75
0xD_1074	IFQEPAR—Inbound frame queue enqueue pointer address register	R/W	0x0000_0000	17.3.3.2.4/17-76

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
RapidIO Doorbell Registers				
0xD_1460	DMR—Doorbell mode register	R/W	0x0000_0000	17.3.3.3.1/17-77
0xD_1464	DSR—Doorbell status register	R/W	0x0000_0000	17.3.3.3.2/17-78
0xD_146C	DQDPAR—Doorbell queue dequeue pointer address register	R/W	0x0000_0000	17.3.3.3.3/17-79
0xD_1474	DQEPAR—Doorbell queue enqueue pointer address register	R/W	0x0000_0000	17.3.3.3.4/17-80
RapidIO Port-Write Registers				
0xD_14E0	PWMR—Port-write mode register	R/W	0x0000_0000	17.3.3.4.1/17-81
0xD_14E4	PWSR—Port-write status register	R/W	0x0000_0000	17.3.3.4.2/17-82
0xD_14EC	PWQBAR—Port-write queue base address register	R/W	0x0000_0000	17.3.3.4.3/17-83
Global Utilities Registers				
Power-On Reset Configuration Values				
0xE_0000	PORPLLSR—POR PLL ratio status register	R	0x00*nn*_00*nn*	18.4.1.1/18-4
0xE_0004	PORBMSR—POR boot mode status register	R	0x*nnnn*_0000	18.4.1.2/18-5
0xE_0008	PORIMPSCR—POR I/O impedance status and control register	R/W	0x000*n*_007F	18.4.1.3/18-6
0xE_000C	PORDEVSR—POR I/O device status register	R	See ref.	18.4.1.4/18-7
0xE_0010	PORDBGMSR—POR debug mode status register	R	See ref.	18.4.1.5/18-9
0xE_0020	GPPORCR—General-purpose POR configuration register	R	See ref.	18.4.1.6/18-9
Signal Multiplexing and GPIO Controls				
0xE_0030	GPIOCR—GPIO control register	R/W	0x0000_0000	18.4.1.7/18-10
0xE_0040	GPOUTDR—General-purpose output data register	R/W	0x0000_0000	18.4.1.8/18-11
0xE_0050	GPINDR—General-purpose input data register	R	0x*nnnn*_0000	18.4.1.9/18-12
0xE_0060	PMUXCR—Alternate function signal multiplex control	R/W	0x0000_0000	18.4.1.9/18-12
Device Disables				
0xE_0070	DEVDISR—Device disable control	R/W	0x0000_0000	18.4.1.11/18-13
Power Management Registers				
0xE_0080	POWMGTCSR—Power management status and control register	R/W	0x0000_0000	18.4.1.12/18-15
Interrupt Reporting				
0xE_0090	MCPSUMR—Machine check summary register	Read/ Clear	0x0000_0000	18.4.1.13/18-16

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
Version Registers				
0xE_00A0	PVR—Processor version register	R	e500 processor version	18.4.1.14/18-17
0xE_00A4	SVR—System version register	R	MPC8540 system version	18.4.1.15/18-18
Debug Control				
0xE_0E00	CLKOCR—Clock out select register	R/W	0x0000_0000	18.4.1.16/18-18
0xE_0E10	DDRDLLCR—DDR DLL control register	R/W	0x0000_0000	18.4.1.17/18-19
0xE_0E20	LBDLLCR—LBC DLL control register	R/W	0x0000_0000	18.4.1.18/18-20
Performance Monitor Control Registers				
0xE_1000	PMGC0—Performance monitor global control register	R/W	0x0000_0000	19.3.2.1/19-5
0xE_1010	PMLCA0—Performance monitor local control register A0	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1014	PMLCB0—Performance monitor local control register B0	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1018	PMC0 (upper)—Performance monitor counter 0 upper	R/W	0x0000_0000	19.3.3.1/19-9
0xE_101C	PMC0 (lower)—Performance monitor counter 0 lower	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1020	PMLCA1—Performance monitor local control register A1	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1024	PMLCB1—Performance monitor local control register B1	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1028	PMC1—Performance monitor counter 1	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1030	PMLCA2—Performance monitor local control register A2	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1034	PMLCB2—Performance monitor local control register B 2	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1038	PMC2—Performance monitor counter 2	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1040	PMLCA3—Performance monitor local control register A3	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1044	PMLCB3—Performance monitor local control register B3	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1048	PMC3—Performance monitor counter 3	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1050	PMLCA4—Performance monitor local control register A4	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1054	PMLCB4—Performance monitor local control register B4	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1058	PMC4—Performance monitor counter 4	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1060	PMLCA5—Performance monitor local control register A5	R/W	0x0000_0000	19.3.2.2/19-5

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xE_1064	PMLCB5—Performance monitor local control register B 5	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1068	PMC5—Performance monitor counter 5	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1070	PMLCA6—Performance monitor local control register A6	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1074	PMLCB6—Performance monitor local control register B6	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1078	PMC6—Performance monitor counter 6	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1080	PMLCA7—Performance monitor local control register A7	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1084	PMLCB7—Performance monitor local control register B7	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1088	PMC7—Performance monitor counter 7	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1090	PMLCA8—Performance monitor local control register A8	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1094	PMLCB8—Performance monitor local control register B8	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1098	PMC8—Performance monitor counter 8	R/W	0x0000_0000	19.3.3.1/19-9
Debug and Watchpoint Monitor Registers				
Watchpoint Monitor Registers				
0xE_2000	WMCR0—Watchpoint monitor control register 0	R/W	0x0000_0000	20.3.1.1/20-11
0xE_2004	WMCR1—Watchpoint monitor control register 1	R/W	0x0000_0000	20.3.1.1/20-11
0xE_200C	WMAR—Watchpoint monitor address register	R/W	0x0000_0000	20.3.1.2/20-13
0xE_2014	WMAMR—Watchpoint monitor address mask register	R/W	0x0000_0000	20.3.1.3/20-14
0xE_2018	WMTMR—Watchpoint monitor transaction mask register	R/W	0x0000_0000	20.3.1.4/20-14
0xE_201C	WMSR—Watchpoint monitor status register	R/W	0x0000_0000	20.3.1.5/20-16
Trace Buffer Registers				
0xE_2040	TBCR0—Trace buffer control register	R/W	0x0000_0000	20.3.2.1/20-16
0xE_2044	TBCR1—Trace buffer control register	R/W	0x0000_0000	20.3.2.1/20-16
0xE_204C	TBAR—Trace buffer address register	R/W	0x0000_0000	20.3.2.2/20-19
0xE_2054	TBAMR—Trace buffer address mask register	R/W	0x0000_0000	20.3.2.3/20-19
0xE_2058	TBTMR—Trace buffer transaction mask register	R/W	0x0000_0000	20.3.2.4/20-20
0xE_205C	TBSR—Trace buffer status register	R/W	0x0000_0000	20.3.2.5/20-21
0xE_2060	TBACR—Trace buffer access control register	R/W	0x0000_0000	20.3.2.6/20-22
0xE_2064	TBADHR—Trace buffer access data high register	R/W	0x0000_0000	20.3.2.7/20-22
0xE_2068	TBADR—Trace buffer access data register	R/W	0x0000_0000	20.3.2.8/20-23

Table 2-9. Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
Context ID Registers				
0xE_20A0	PCIDR—Programmed context ID register	R/W	0x0000_0000	20.3.3.1/20-24
0xE_20A4	CCIDR—Current context ID register	R/W	0x0000_0000	20.3.3.2/20-24
Other Registers				
0xE_20B0	TOSR—Trigger output source register	R/W	0x0000_0000	20.3.4.1/20-25

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 MPC8540 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 MPC8540 signals are grouped as follows:

- DDR memory interface signals
- RapidIO interface signals
- PCI/PCI-X interface signals
- Ethernet management interface signals
- TSEC1 interface signals
- TSEC2 interface signals
- FEC10/100 interface signals
- Local bus interface signals
- DMA interface signals
- PIC interface signals

- DUART interface signals
- I^2C 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 MPC8540, showing how the signals are grouped. Refer to the *MPC8540 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.

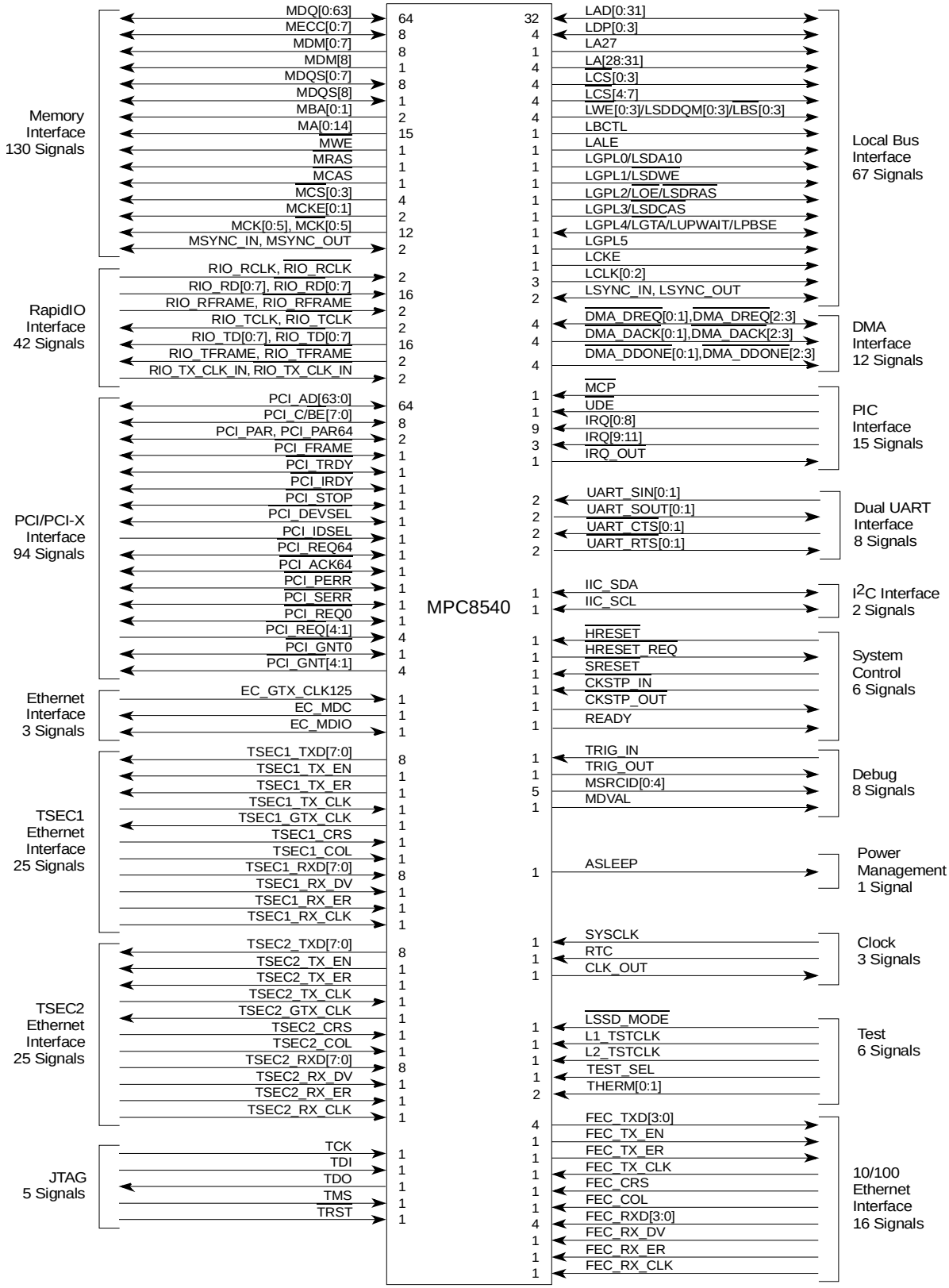


Figure 3-1. MPC8540 Signal Groupings

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. MPC8540 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	17-5/17-8
RIO_RD[0:7], $\overline{\text{RIO\_RD}}$[0:7]	RapidIO receive data	RapidIO	—	16	I	17-5/17-8
RIO_RFRAME, $\overline{\text{RIO\_RFRAME}}$	RapidIO receive frame	RapidIO	—	2	I	17-5/17-8
RIO_TCLK, $\overline{\text{RIO\_TCLK}}$	RapidIO transmit clock	RapidIO	—	2	O	17-5/17-8

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
RIO_TD[0:7], RIO_TD[0:7]	RapidIO transmit data	RapidIO	—	16	O	17-5/17-8
RIO_TFRAME, RIO_TFRAME	RapidIO transmit frame	RapidIO	—	2	O	17-5/17-8
RIO_TX_CLK_IN RIO_TX_CLK_IN	RapidIO transmit clock in	RapidIO	—	2	I	17-5/17-8
PCI_AD[63:0]	PCI address/data	PCI/X	—	64	I/O	16-2/16-8
PCI_C/BE[7:0]	PCI command/byte enable	PCI/X	—	8	I/O	16-2/16-8
PCI_PAR	PCI parity	PCI/X	—	1	I/O	16-2/16-8
PCI_PAR64	PCI parity 64	PCI/X	—	1	I/O	16-2/16-8
PCI_FRAME	PCI frame	PCI/X	—	1	I/O	16-2/16-8
PCI_TRDY	PCI target ready	PCI/X	—	1	I/O	16-2/16-8
PCI_IRDY	PCI initiator ready	PCI/X	—	1	I/O	16-2/16-8
PCI_STOP	PCI stop	PCI/X	—	1	I/O	16-2/16-8
PCI_DEVSEL	PCI device select	PCI/X	—	1	I/O	16-2/16-8
PCI_IDSEL	PCI initial device select	PCI/X	—	1	I	16-2/16-8
PCI_REQ64	PCI request 64	PCI/X	cfg_pci_width	1	I/O	16-2/16-8
PCI_ACK64	PCI acknowledge 64	PCI/X	—	1	I/O	16-2/16-8
PCI_PERR	PCI parity error	PCI/X	—	1	I/O	16-2/16-8
PCI_SERR	PCI system error	PCI/X	—	1	I/O	16-2/16-8
PCI_REQ0	PCI request 0	PCI/X	—	1	I/O	16-2/16-8
PCI_REQ[4:1]	PCI request 4–1	PCI/X	—	4	I	16-2/16-8
PCI_GNT0	PCI grant 0	PCI/X	—	1	I/O	16-2/16-8
PCI_GNT[4:1]	PCI grant 4–1	PCI/X	cfg_pci_mode cfg_pci_debug cfg_pci_arbiter cfg_pci_impd	4	O	16-2/16-8
EC_GTX_CLK125	Gigabit reference clock	Gigabit clock	—	1	I	14-3/14-14
EC_MDC	Ethernet management data clock	Ethernet management	cfg_tsec_reduce	1	O	14-3/14-14
EC_MDIO	Ethernet management data in/out	Ethernet management	—	1	I/O	14-3/14-14
TSEC1_TXD7	TSEC1 transmit data 7	TSEC1	cfg_tsec1	1	O	14-3/14-14
TSEC1_TXD[6:4]	TSEC1 transmit data 6–4	TSEC1	cfg_rom_loc[0:2]	3	O	14-3/14-14

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
TSEC1_TXD3	TSEC1 transmit data 3	TSEC1	—	1	O	14-3/14-14
TSEC1_TXD2	TSEC1 transmit data 2	TSEC1	—	1	O	14-3/14-14
TSEC1_TXD1	TSEC1 transmit data 1	TSEC1	—	1	O	14-3/14-14
TSEC1_TXD0	TSEC1 transmit data 0	TSEC1	—	1	O	14-3/14-14
TSEC1_TX_EN	TSEC1 transmit enable	TSEC1	—	1	O	14-3/14-14
TSEC1_TX_ER	TSEC1 transmit error	TSEC1	—	1	O	14-3/14-14
TSEC1_TX_CLK	TSEC1 transmit clock in	TSEC1	—	1	I	14-3/14-14
TSEC1_GTX_CLK	TSEC1 transmit clock out	TSEC1	_	1	O	14-3/14-14
TSEC1_CRS	TSEC1 carrier sense	TSEC1	—	1	I	14-3/14-14
TSEC1_COL	TSEC1 collision detect	TSEC1	—	1	I	14-3/14-14
TSEC1_RXD[7:0]	TSEC1 receive data	TSEC1	—	8	I	14-3/14-14
TSEC1_RX_DV	TSEC1 receive data valid	TSEC1	—	1	I	14-3/14-14
TSEC1_RX_ER	TSEC1 receiver error	TSEC1	—	1	I	14-3/14-14
TSEC1_RX_CLK	TSEC1 receive clock	TSEC1	—	1	I	14-3/14-14
TSEC2_TXD7	TSEC2 transmit data 7	TSEC2	cfg_tsec2	1	O	14-3/14-14
TSEC2_TXD[6:5]	TSEC2 transmit data 6–5	TSEC2	cfg_lb_hold[0:1]	2	O	14-3/14-14
TSEC2_TXD[4:2]	TSEC2 transmit data 4–2	TSEC2	cfg_dev_ID[7:5]	3	O	14-3/14-14
TSEC2_TXD1	TSEC2 transmit data 1	TSEC2	—	1	O	14-3/14-14
TSEC2_TXD0	TSEC2 transmit data 0	TSEC2	—	1	O	14-3/14-14
TSEC2_TX_EN	TSEC2 transmit enable	TSEC2	—	1	O	14-3/14-14
TSEC2_TX_ER	TSEC2 transmit error	TSEC2	—	1	O	14-3/14-14
TSEC2_TX_CLK	TSEC2 transmit clock in	TSEC2	—	1	I	14-3/14-14
TSEC2_GTX_CLK	TSEC2 transmit clock out	TSEC2	—	1	O	14-3/14-14
TSEC2_CRS	TSEC2 carrier sense	TSEC2	—	1	I	14-3/14-14
TSEC2_COL	TSEC2 collision detect	TSEC2	—	1	I	14-3/14-14
TSEC2_RXD[0:7]	TSEC2 receive data	TSEC2	—	8	I	14-3/14-14
TSEC2_RX_DV	TSEC2 receive data valid	TSEC2	—	1	I	14-3/14-14
TSEC2_RX_ER	TSEC2 receiver error	TSEC2	—	1	I	14-3/14-14
TSEC2_RX_CLK	TSEC2 receive clock	TSEC2	—	1	I	14-3/14-14
FEC_TXD[0:3]	FEC transmit data	10/100	—	4	O	21-1/21-8
FEC_TX_EN	FEC transmit enable	10/100	—	1	O	21-1/21-8
FEC_TX_ER	FEC transmit error	10/100	—	1	O	21-1/21-8

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
FEC_TX_CLK	FEC transmit clock in	10/100	—	1	I	21-1/21-8
FEC_CRS	FEC carrier sense	10/100	—	1	I	21-1/21-8
FEC_COL	FEC collision detect	10/100	—	1	I	21-1/21-8
FEC_RXD[0:3]	FEC receive data	10/100	—	4	I	21-1/21-8
FEC_RX_DV	FEC receive data valid	10/100	—	1	I	21-1/21-8
FEC_RX_ER	FEC receiver error	10/100	—	1	I	21-1/21-8
FEC_RX_CLK	FEC receive clock	10/100	—	1	I	21-1/21-8
LAD[0:31]	LBC address/data	LBC	cfg_gpporcr	32	I/O	13-1/13-5
LDP[0:3]	LBC data parity	LBC	—	4	I/O	13-1/13-5
LA27	LBC burst address	LBC	cfg_cpu_boot	1	O	13-1/13-5
LA[28:31]	LBC port address	LBC	cfg_sys_pll0 cfg_sys_pll1 cfg_sys_pll2 cfg_sys_pll3	4	O	13-1/13-5
$\overline{\text{LCS}}$[0:4]	LBC chip select 0–4	LBC	—	5	O	13-1/13-5
$\overline{\text{LCS5}}$	LBC chip select 5	LBC	$\overline{\text{DMA\_DREQ2}}$	1	O	13-1/13-5
$\overline{\text{LCS6}}$	LBC chip select 6	LBC	$\overline{\text{DMA\_DACK2}}$	1	O	13-1/13-5
$\overline{\text{LCS7}}$	LBC chip select 7	LBC	$\overline{\text{DMA\_DDONE2}}$	1	O	13-1/13-5
$\overline{\text{LWE}}$[0:1]/ LSDDQM[0:1]/ $\overline{\text{LBS}}$[0:1]	LBC write enable/byte lane data mask/byte select 0–1	LBC	cfg_pci_hold[0:1]	2	O	13-1/13-5
$\overline{\text{LWE}}$[2:3]/ LSDDQM[2:3]/ $\overline{\text{LBS}}$[2:3]	LBC write enable/byte lane data mask/byte select 2–3	LBC	cfg_host_agt0 cfg_host_agt1	2	O	13-1/13-5
LBCTL	LBC data buffer control	LBC	—	1	O	13-1/13-5
LALE	LBC address latch enable	LBC	cfg_core_pll0	1	O	13-1/13-5
LGPL0/LSDA10	LBC UPM general purpose line 0/SDRAM address bit 10	LBC	cfg_rio_clk0	1	O	13-1/13-5
LGPL1/$\overline{\text{LSDWE}}$	LBC GP line 1/SDRAM write enable	LBC	cfg_rio_clk1	1	O	13-1/13-5
LGPL2/$\overline{\text{LOE}}$/ $\overline{\text{LSDRAS}}$	LBC GP line 2/output enable/SDRAM RAS	LBC	cfg_core_pll1	1	O	13-1/13-5
LGPL3/$\overline{\text{LSDCAS}}$	LBC GP line 3/SDRAM CAS	LBC	cfg_boot_seq0	1	O	13-1/13-5

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
LGPL4/LGTA/ LUPWAIT/LPBSE	LBC GP line 4/GPCM terminate access/UPM wait/parity byte select	LBC	—	1	I/O	13-1/13-5
LGPL5	LBC GP line 5 address	LBC	cfg_boot_seq1	1	O	13-1/13-5
LCKE	LBC clock enable	LBC	—	1	O	13-1/13-5
LCLK[0:2]	LBC clock	LBC	—	3	O	13-1/13-5
LSYNC_IN, LSYNC_OUT	LBC DLL synchronization	LBC	—	2	I/O	13-1/13-5
DMA_DREQ[0:1]	DMA request 0–1	DMA	—	2	I	15-3/15-6
DMA_DREQ2	DMA request 2	DMA	LCS5	1	I	15-3/15-6
DMA_DREQ3	DMA request 3	DMA	IRQ9	1	I	15-3/15-6
DMA_DACK[0:1]	DMA acknowledge 0–1	DMA	—	2	O	15-3/15-6
DMA_DACK2	DMA acknowledge 2	DMA	LCS6	1	O	15-3/15-6
DMA_DACK3	DMA acknowledge 3	DMA	IRQ10	1	O	15-3/15-6
DMA_DDONE[0:1]	DMA done 0–1	DMA	—	2	O	15-3/15-6
DMA_DDONE2	DMA done 2	DMA	LCS7	1	O	15-3/15-6
DMA_DDONE3	DMA done 3	DMA	IRQ11	1	O	15-3/15-6
MCP	Machine check processor	PIC	—	1	I	10-6/10-8
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	DMA_DREQ3	1	I	10-6/10-8
IRQ10	External interrupt 10	PIC	DMA_DACK3	1	I	10-6/10-8
IRQ11	External interrupt 11	PIC	DMA_DDONE3	1	I	10-6/10-8
IRQ_OUT	Interrupt output	PIC	—	1	O	10-6/10-8
UART_SIN[0:1]	DUART serial data in	Dual UART	—	2	I	12-2/12-4
UART_SOUT[0:1]	DUART serial data out	Dual UART	—	2	O	12-2/12-4
UART_CTS[0:1]	DUART clear to send	Dual UART	—	2	I	12-2/12-4
UART_RTS[0:1]	DUART ready to send	Dual UART	—	2	O	12-2/12-4
IIC_SDA	I^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
HRESET	Hard reset	System control	—	1	I	4-2/4-2
HRESET_REQ	Hard reset request	System control	—	1	O	4-2/4-2

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{SRESET}}$	Soft reset	System control	—	1	I	4-2/4-2
$\overline{\text{CKSTP\_IN}}$	Checkstop in	System control	—	1	I	18.2/18-1
$\overline{\text{CKSTP\_OUT}}$	Checkstop out	System control	—	1	O	18.2/18-1
READY	Device ready	System control	TRIG_OUT	1	O	4-2/4-2
TRIG_IN	Watchpoint trigger in	Debug	—	1	I	20-2/20-6
TRIG_OUT	Watchpoint trigger out	Debug	READY	1	O	20-2/20-6
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	20-2/20-6
MDVAL	Memory debug data valid	Debug	—	1	O	20-2/20-6
ASLEEP	Asleep	Power mgmt	—	1	O	18.2/18-1
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	18.2/18-1
$\overline{\text{LSSD\_MODE}}$	LSSD mode	Test	—	1	I	20-2/20-6
L1_TSTCLK	L1 test clock	Test	—	1	I	20-2/20-6
L2_TSTCLK	L2 test clock	Test	—	1	I	20-2/20-6
$\overline{\text{TEST\_SEL}}$	Test select	Test	—	1	I	20-2/20-6
THERM[0:1]	Thermal resistor access	Test	—	2	I	20-2/20-6
TCK	Test clock	JTAG	—	1	I	20-2/20-6
TDI	Test data in	JTAG	—	1	I	20-2/20-6
TDO	Test data out	JTAG	—	1	O	20-2/20-6
TMS	Test mode select	JTAG	—	1	I	20-2/20-6
$\overline{\text{TRST}}$	Test reset	JTAG	—	1	I	20-2/20-6

Table 3-2. MPC8540 Alphabetical Signal Reference

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
ASLEEP	Asleep	Power mgmt	—	1	O	18.2/18-1
$\overline{\text{CKSTP\_IN}}$	Checkstop in	System control	—	1	I	18.2/18-1
$\overline{\text{CKSTP\_OUT}}$	Checkstop out	System control	—	1	O	18.2/18-1
CLK_OUT	Clock out	Clock	—	1	O	18.2/18-1

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{DMA\_DACK2}}$	DMA acknowledge 2	DMA	$\overline{\text{LCS6}}$	1	O	15-3/15-6
$\overline{\text{DMA\_DACK3}}$	DMA acknowledge 3	DMA	IRQ10	1	O	15-3/15-6
$\overline{\text{DMA\_DACK}}$[0:1]	DMA acknowledge 0–1	DMA	—	2	O	15-3/15-6
$\overline{\text{DMA\_DDONE2}}$	DMA done 2	DMA	$\overline{\text{LCS7}}$	1	O	15-3/15-6
$\overline{\text{DMA\_DDONE3}}$	DMA done 3	DMA	IRQ11	1	O	15-3/15-6
$\overline{\text{DMA\_DDONE}}$[0:1]	DMA done 0–1	DMA	—	2	O	15-3/15-6
$\overline{\text{DMA\_DREQ2}}$	DMA request 2	DMA	$\overline{\text{LCS5}}$	1	I	15-3/15-6
$\overline{\text{DMA\_DREQ3}}$	DMA request 3	DMA	IRQ9	1	I	15-3/15-6
$\overline{\text{DMA\_DREQ}}$[0:1]	DMA request 0–1	DMA	—	2	I	15-3/15-6
EC_GTX_CLK125	Gigabit reference clock	Gigabit clock	—	1	I	14-3/14-14
EC_MDC	Ethernet management data clock	Ethernet management	cfg_tsec_reduce	1	O	14-3/14-14
EC_MDIO	Ethernet management data in/out	Ethernet management	—	1	I/O	14-3/14-14
FEC_COL	FEC collision detect	10/100	—	1	I	21-1/21-8
FEC_CRS	FEC carrier sense	10/100	—	1	I	21-1/21-8
FEC_RXD[0:3]	FEC receive data	10/100	—	4	I	21-1/21-8
FEC_RX_CLK	FEC receive clock	10/100	—	1	I	21-1/21-8
FEC_RX_DV	FEC receive data valid	10/100	—	1	I	21-1/21-8
FEC_RX_ER	FEC receiver error	10/100	—	1	I	21-1/21-8
FEC_TXD[0:3]	FEC transmit data	10/100	—	4	O	21-1/21-8
FEC_TX_CLK	FEC transmit clock in	10/100	—	1	I	21-1/21-8
FEC_TX_EN	FEC transmit enable	10/100	—	1	O	21-1/21-8
FEC_TX_ER	FEC transmit error	10/100	—	1	O	21-1/21-8
$\overline{\text{HRESET}}$	Hard reset	System control	—	1	I	4-2/4-2
$\overline{\text{HRESET\_REQ}}$	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

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
IRQ_OUT	Interrupt output	PIC	—	1	O	10-6/10-8
L1_TSTCLK	L1 test clock	Test	—	1	I	20-2/20-6
L2_TSTCLK	L2 test clock	Test	—	1	I	20-2/20-6
LA27	LBC burst address	LBC	cfg_cpu_boot	1	O	13-1/13-5
LAD[0:31]	LBC address/data	LBC	cfg_gpporcr	32	I/O	13-1/13-5
LALE	LBC address latch enable	LBC	cfg_core_pll0	1	O	13-1/13-5
LA[28:31]	LBC port address	LBC	cfg_sys_pll0 cfg_sys_pll1 cfg_sys_pll2 cfg_sys_pll3	4	O	13-1/13-5
LBCTL	LBC data buffer control	LBC	—	1	O	13-1/13-5
LCKE	LBC clock enable	LBC	—	1	O	13-1/13-5
LCLK[0:2]	LBC clock	LBC	—	3	O	13-1/13-5
LCS5	LBC chip select 5	LBC	DMA_DREQ2	1	O	13-1/13-5
LCS6	LBC chip select 6	LBC	DMA_DACK2	1	O	13-1/13-5
LCS7	LBC chip select 7	LBC	DMA_DDONE2	1	O	13-1/13-5
LCS[0:4]	LBC Chip select 0–4	LBC	—	5	O	13-1/13-5
LDP[0:3]	LBC data parity	LBC	—	4	I/O	13-1/13-5
LGPL0/LSDA10	LBC UPM general purpose line 0/SDRAM address bit 10	LBC	cfg_rio_clk0	1	O	13-1/13-5
LGPL1/LSDWE	LBC GP line 1/SDRAM write enable	LBC	cfg_rio_clk1	1	O	13-1/13-5
LGPL2/LOE/ LSDRAS	LBC GP line 2/output enable/SDRAM RAS	LBC	cfg_core_pll1	1	O	13-1/13-5
LGPL3/LSDCAS	LBC GP line 3/SDRAM CAS	LBC	cfg_boot_seq0	1	O	13-1/13-5
LGPL4/LGTA/ LUPWAIT/LPBSE	LBC GP line 4/GPCM terminate access/UPM wait/parity byte select	LBC	—	1	I/O	13-1/13-5
LGPL5	LBC GP line 5 address	LBC	cfg_boot_seq1	1	O	13-1/13-5
LSSD_MODE	LSSD mode	Test	—	1	I	20-2/20-6
LSYNC_IN, LSYNC_OUT	LBC DLL synchronization	LBC	—	2	I/O	13-1/13-5
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	13-1/13-5

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{LWE}}$[2:3]/ LSDDQM[2:3]/ $\overline{\text{LBS}}$[2:3]	LBC write enable/byte lane data mask/byte select 2–3	LBC	cfg_host_agt0 cfg_host_agt1	2	O	13-1/13-5
MA[0:14]	DDR address	DDR memory	—	15	O	9-3/9-5
MBA[0:1]	DDR bank select	DDR memory	—	2	O	9-3/9-5
$\overline{\text{MCAS}}$	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], $\overline{\text{MCK}}$[0:5]	DDR differential clocks (3 pairs/DIMM)	DDR memory	—	12	O	9-4/9-8
$\overline{\text{MCP}}$	Machine check processor	PIC	—	1	I	10-6/10-8
$\overline{\text{MCS}}$[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	20-2/20-6
MECC[0:7]	DDR error correcting code	DDR memory	—	8	I/O	9-3/9-5
$\overline{\text{MRAS}}$	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
MSRCID[2:4]	Memory debug source port ID 2–4	Debug	—	3	O	20-2/20-6
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
$\overline{\text{PCI\_ACK64}}$	PCI acknowledge 64	PCI/X	—	1	I/O	16-2/16-8
PCI_AD[63:0]	PCI address/data	PCI/X	—	64	I/O	16-2/16-8
PCI_C/$\overline{\text{BE}}$[7:0]	PCI command/byte enable	PCI/X	—	8	I/O	16-2/16-8
$\overline{\text{PCI\_DEVSEL}}$	PCI device select	PCI/X	—	1	I/O	16-2/16-8
$\overline{\text{PCI\_FRAME}}$	PCI frame	PCI/X	—	1	I/O	16-2/16-8
$\overline{\text{PCI\_GNT0}}$	PCI grant 0	PCI/X	—	1	I/O	16-2/16-8

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
PCI_GNT[4:1]	PCI grant 4–1	PCI/X	cfg_pci_mode cfg_pci_debug cfg_pci_arbiter cfg_pci_impd	4	O	16-2/16-8
PCI_IDSEL	PCI initial device select	PCI/X	—	1	I	16-2/16-8
PCI_IRDY	PCI initiator ready	PCI/X	—	1	I/O	16-2/16-8
PCI_PAR	PCI parity	PCI/X	—	1	I/O	16-2/16-8
PCI_PAR64	PCI parity 64	PCI/X	—	1	I/O	16-2/16-8
PCI_PERR	PCI parity error	PCI/X	—	1	I/O	16-2/16-8
PCI_REQ0	PCI request 0	PCI/X	—	1	I/O	16-2/16-8
PCI_REQ64	PCI request 64	PCI/X	cfg_pci_width	1	I/O	16-2/16-8
PCI_REQ[4:1]	PCI request 4–1	PCI/X	—	4	I	16-2/16-8
PCI_SERR	PCI system error	PCI/X	—	1	I/O	16-2/16-8
PCI_STOP	PCI stop	PCI/X	—	1	I/O	16-2/16-8
PCI_TRDY	PCI target ready	PCI/X	—	1	I/O	16-2/16-8
READY	Device ready	System control	TRIG_OUT	1	O	4-2/4-2
RIO_RCLK, RIO_RCLK	RapidIO receive clocks	RapidIO	—	2	I	17-5/17-8
RIO_RD[0:7], RIO_RD[0:7]	RapidIO receive data	RapidIO	—	16	I	17-5/17-8
RIO_RFRAME, RIO_RFRAME	RapidIO receive frame	RapidIO	—	2	I	17-5/17-8
RIO_TCLK, RIO_TCLK	RapidIO transmit clock	RapidIO	—	2	O	17-5/17-8
RIO_TD[0:7], RIO_TD[0:7]	RapidIO transmit data	RapidIO	—	16	O	17-5/17-8
RIO_TFRAME, RIO_TFRAME	RapidIO transmit frame	RapidIO	—	2	O	17-5/17-8
RIO_TX_CLK_IN RIO_TX_CLK_IN	RapidIO transmit clock in	RapidIO	—	2	I	17-5/17-8
RTC	Real time clock	Clock	—	1	I	4-3/4-3
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	20-2/20-6
TDI	Test data in	JTAG	—	1	I	20-2/20-6
TDO	Test data out	JTAG	—	1	O	20-2/20-6

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
$\overline{\text{TEST\_SEL}}$	Test select	Test	—	1	I	20-2/20-6
THERM[0:1]	Thermal resistor access	Test	—	2	I	20-2/20-6
TMS	Test mode select	JTAG	—	1	I	20-2/20-6
TRIG_IN	Watchpoint trigger in	Debug	—	1	I	20-2/20-6
TRIG_OUT	Watchpoint trigger out	Debug	READY	1	O	20-2/20-6
$\overline{\text{TRST}}$	Test reset	JTAG	—	1	I	20-2/20-6
TSEC1_COL	TSEC1 collision detect	TSEC1	—	1	I	14-3/14-14
TSEC1_CRS	TSEC1 carrier sense	TSEC1	—	1	I	14-3/14-14
TSEC1_GTX_CLK	TSEC1 transmit clock out	TSEC1	—	1	O	14-3/14-14
TSEC1_RXD[7:0]	TSEC1 receive data	TSEC1	—	8	I	14-3/14-14
TSEC1_RX_CLK	TSEC1 receive clock	TSEC1	—	1	I	14-3/14-14
TSEC1_RX_DV	TSEC1 receive data valid	TSEC1	—	1	I	14-3/14-14
TSEC1_RX_ER	TSEC1 receiver error	TSEC1	—	1	I	14-3/14-14
TSEC1_TXD0	TSEC1 transmit data 0	TSEC1	—	1	O	14-3/14-14
TSEC1_TXD1	TSEC1 transmit data 1	TSEC1	—	1	O	14-3/14-14
TSEC1_TXD2	TSEC1 transmit data 2	TSEC1	—	1	O	14-3/14-14
TSEC1_TXD3	TSEC1 transmit data 3	TSEC1	—	1	O	14-3/14-14
TSEC1_TXD[6:4]	TSEC1 transmit data 6–4	TSEC1	cfg_rom_loc[0:2]	3	O	14-3/14-14
TSEC1_TXD7	TSEC1 transmit data 7	TSEC1	cfg_tsec1	1	O	14-3/14-14
TSEC1_TX_CLK	TSEC1 transmit clock in	TSEC1	—	1	I	14-3/14-14
TSEC1_TX_EN	TSEC1 transmit enable	TSEC1	—	1	O	14-3/14-14
TSEC1_TX_ER	TSEC1 transmit error	TSEC1	—	1	O	14-3/14-14
TSEC2_COL	TSEC2 collision detect	TSEC2	—	1	I	14-3/14-14
TSEC2_CRS	TSEC2 carrier sense	TSEC2	—	1	I	14-3/14-14
TSEC2_GTX_CLK	TSEC2 transmit clock out	TSEC2	—	1	O	14-3/14-14
TSEC2_RXD[0:7]	TSEC2 receive data	TSEC2	—	8	I	14-3/14-14
TSEC2_RX_CLK	TSEC2 receive clock	TSEC2	—	1	I	14-3/14-14
TSEC2_RX_DV	TSEC2 receive data valid	TSEC2	—	1	I	14-3/14-14
TSEC2_RX_ER	TSEC2 receiver error	TSEC2	—	1	I	14-3/14-14
TSEC2_TXD0	TSEC2 transmit data 0	TSEC2	—	1	O	14-3/14-14
TSEC2_TXD1	TSEC2 transmit data 1	TSEC2	—	1	O	14-3/14-14
TSEC2_TXD[4:2]	TSEC2 transmit data 4–2	TSEC2	cfg_dev_ID[7:5]	3	O	14-3/14-14

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

Name	Description	Functional Block	Alternate Function(s)	No. of Signals	I/O	Table/ Page
TSEC2_TXD[6:5]	TSEC2 transmit data 6–5	TSEC2	cfg_lb_hold[0:1]	2	O	14-3/14-14
TSEC2_TXD7	TSEC2 transmit data 7	TSEC2	cfg_tsec2	1	O	14-3/14-14
TSEC2_TX_CLK	TSEC2 transmit clock in	TSEC2	—	1	I	14-3/14-14
TSEC2_TX_EN	TSEC2 transmit enable	TSEC2	—	1	O	14-3/14-14
TSEC2_TX_ER	TSEC2 transmit error	TSEC2	—	1	O	14-3/14-14
$\overline{\text{UART\_CTS}}$[0:1]	DUART clear to send	Dual UART	—	2	I	12-2/12-4
$\overline{\text{UART\_RTS}}$[0:1]	DUART ready to send	Dual UART	—	2	O	12-2/12-4
UART_SIN[0:1]	DUART serial data in	Dual UART	—	2	I	12-2/12-4
UART_SOUT[0:1]	DUART serial data out	Dual UART	—	2	O	12-2/12-4
$\overline{\text{UDE}}$	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 *MPC8540 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 *MPC8540 Integrated Processor Hardware Specifications*.

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

Table 3-3. MPC8540 Reset Configuration Signals

Functional Interface	Functional Signal Name	Reset Configuration Name	Default
PCI	PCI_REQ64	cfg_pci_width	1
	PCI_GNT4	cfg_pci_mode	1
	PCI_GNT3	cfg_pci_debug	1
	PCI_GNT2	cfg_pci_arbiter	1
	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
LBC	LA27	cfg_cpu_boot	1
	LA[28:31]	cfg_sys_pll[0:3]	Must be driven
	LWE[0:1]/LBS[0:1]	cfg_pci_hold[0:1]	11
	LWE[2:3]/LBS[2:3]	cfg_host_agt0 cfg_host_agt1	11
	LALE	cfg_core_pll0	Must be driven
	LGPL0/LSDA10	cfg_rio_clk0	1
	LGPL1/LSDWE	cfg_rio_clk1	1
	LGPL2/LOE/LSDRAS	cfg_core_pll1	Must be driven
	LGPL3/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 (HRESET is asserted), the MPC8540 aborts all current internal and external transactions and releases all bidirectional I/O signals to a high-impedance state. See Chapter 4, “Reset, Clocking, and Initialization,” for a complete description of the reset functionality.

During reset, the MPC8540 ignores most input signals (except for the reset configuration signals) and drives most of the output-only signals to an inactive state. Table 3-4 shows the states of the output-only signals (that is, the signals that are not multiplexed with other inputs, or are not used as reset configuration signals during system reset).

Table 3-4. Output Signal States During System Reset

Interface	Signal	State During Reset
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
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
FEC	FEC_TXD[0:3]	High-Z
FEC	FEC_TX_EN	Driven Low
FEC	FEC_TX_ER	High-Z
LBC	LCLK[0:2]	High-Z
LBC	$\overline{\text{LCS}}$[0:5]	High-Z
LBC	$\overline{\text{DMA\_DACK2}}$/$\overline{\text{LCS6}}$ $\overline{\text{DMA\_DDONE2}}$/$\overline{\text{LCS7}}$	High-Z
LBC	LBCTL	Input—reset config (test only)
DMA	$\overline{\text{DMA\_DACK}}$[0:1]	High-Z

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

Interface	Signal	State During Reset
DMA	DMA_DACK2/LCS6 DMA_DDONE2/LCS7	High-Z
DMA	DMA_DDONE0	High-Z
DMA	DMA_DDONE1	Driven (test only)
PIC	IRQ8	Driven (test only)
PIC	IRQ_OUT	High-Z
Dual UART	UART_SOUT[0:1]	High-Z
Dual UART	UART_RTS[0:1]	High-Z
System Control	HRESET_REQ	High-Z
System Control	CKSTP_OUT	High-Z
Debug	TRIG_OUT/READY	Input—reset config (test only)
Debug	MSRCID[2:4]	High-Z
Debug	MDVAL	High-Z
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 MPC8540, 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 MPC8540. 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	4.4.1.1/4-9
READY	O	The MPC8540 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 MPC8540	4.4.4.1/4-23
RTC	I	Real time clock input	4.4.4.4/4-25

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 MPC8540. 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 18, "Global Utilities."

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

Signal	I/O	Description	
HRESET	I	Hard reset. Causes the MPC8540 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 MPC8540 signals during reset.
		Timing	Assertion/Negation—The *MPC8540 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 MPC8540 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. 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. 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 20, "Debug Features and Watchpoint Facility," for more information on TOSR and TRIG_OUT.	
		State Meaning	Asserted—Indicates that the MPC8540 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 MPC8540. Note that some clock signals are specific to blocks within the MPC8540, 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 MPC8540; the signal driven on the CLK_OUT pin is selectable and described in Section 18.4.1.16, "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 MPC8540. 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 MPC8540 is in agent or host mode. The MPC8540 does not provide a separate PCI_CLK output in host mode.	
		Timing	Assertion/Negation—See the *MPC8540 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 *MPC8540 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 MPC8540 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.

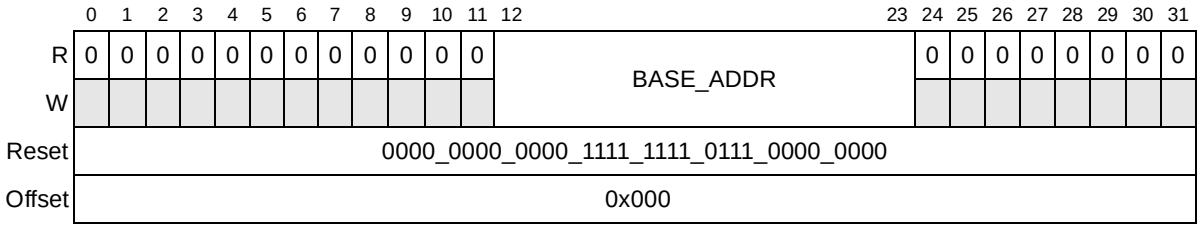


Figure 4-1. Configuration, Control, and Status Register Base Address Register (CCSRBAR)

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.

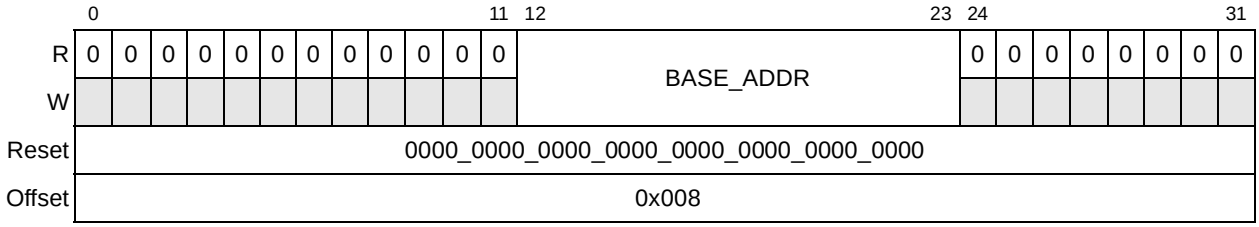


Figure 4-2. Alternate Configuration Base Address Register (ALTCBAR)

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.

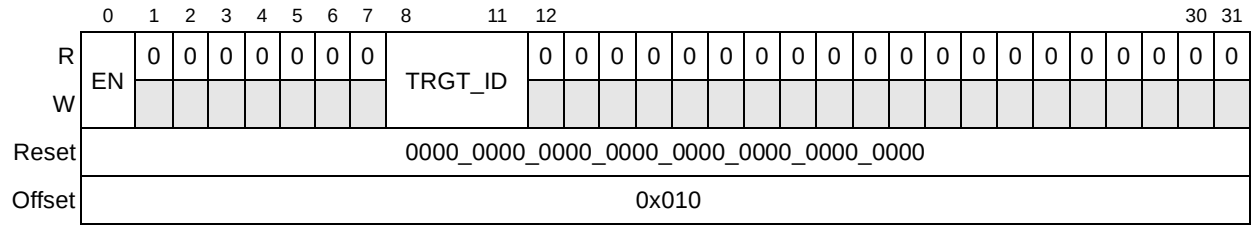


Figure 4-3. Alternate Configuration Attribute Register (ALTCAR)

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 0101–0111Reserved 1000 Configuration, control, and status registers 1001–1011Reserved 1100 RapidIO 1101 Reserved 1110 Reserved 1111 Local memory —DDR SDRAM and on-chip SRAM
12–31	—	Write reserved, read = 0

4.3.1.3 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 MPC8540 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 MPC8540 is configured to be in boot holdoff mode. If translation is performed to a page outside of the default boot ROM address range defined in the MPC8540 (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.

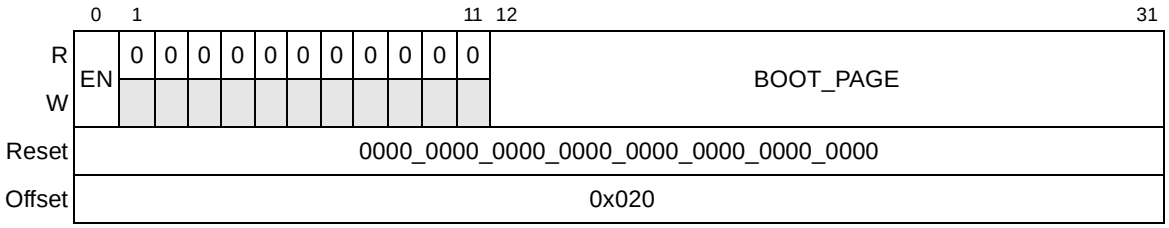


Figure 4-4. Boot Page Translation Register (BPTR)

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 MPC8540 device, the POR configurations, and the clocking on the MPC8540.

4.4.1 Reset Operations

The MPC8540 has reset input signals for hard and soft reset operation.

4.4.1.1 Soft Reset

Assertion of $\overline{\text{SRESET}}$ causes a machine check interrupt to 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 18.4.1.13, "Machine Check Summary Register (MCPSUMR)," for more information on the setting of the soft reset flag. Note that if $\overline{\text{SRESET}}$ is asserted before the e500 core is configured to handle a machine check interrupt, a core checkstop condition occurs, which causes $\overline{\text{CKSTP\_OUT}}$ to assert.

4.4.1.2 Hard Reset

The MPC8540 can be completely reset by the assertion of the $\overline{\text{HRESET}}$ input. The assertion of this signal by external logic is the equivalent of a POR and causes the sequence of events described in Section 4.4.2, "Power-On Reset Sequence."

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

The MPC8540 hard reset request output signal ($\overline{\text{HRESET\_REQ}}$) 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 17.4.1, "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 MPC8540 is as follows:

1. Power is applied to meet the specifications in the *MPC8540 Integrated Processor Hardware Specifications*.
2. System asserts $\overline{\text{HRESET}}$ and $\overline{\text{TRST}}$ causing all registers to be initialized to their default states and most I/O drivers to be 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 $\overline{\text{HRESET}}$ 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 MPC8540 enables I/O drivers.
6. The MPC8540 PCI/PCI-X interface can assert $\overline{\text{DEVSEL}}$ in response to configuration cycles.
7. The e500 PLL configuration inputs are applied, allowing the e500 PLL to begin locking to the device clock (the CCB clock).
8. The CCB clock is cycled for approximately 50 µs to lock the e500 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 MPC8540 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 20.3.4.1, “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 20.3.4, “Trigger Out Function.” For more information about power management states, see Section 18.4.1, “Register Descriptions.”

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

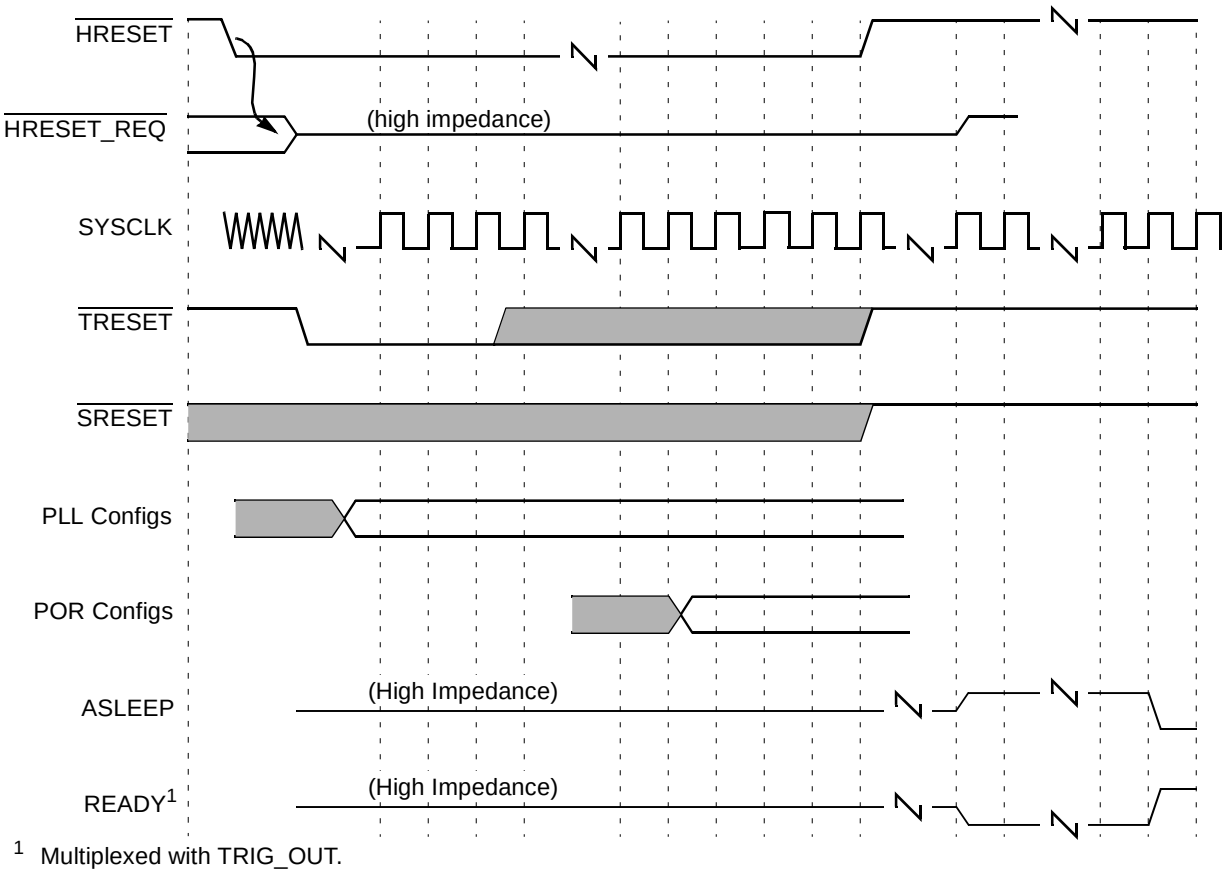


Figure 4-5. Power-On Reset Sequence

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 *MPC8540Integrated 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 18, “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 MPC8540. 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 18.4.1.1, “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 18.4.1.1, "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 MPC8540 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 18.4.1.2, "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 MPC8540 to act as a host or as an agent of a master on another interface. In host mode, the MPC8540 is immediately enabled to master transactions to the RapidIO and PCI interfaces. If the MPC8540 is an agent on the RapidIO or the PCI interfaces, then the MPC8540 is disabled from mastering transactions on that interface until the external host enables it to do so. The external host does this by setting the control registers of the MPC8540's interfaces appropriately. See details in the PCI and RapidIO programming models described in Chapter 16, "PCI/PCI-X Bus Interface," and Chapter 17, "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 18.4.1.2, "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	MPC8540 acts as an agent of both a PCI/PCI-X and a RapidIO device.
		01	MPC8540 acts as an agent of a RapidIO host.
		10	MPC8540 acts as an agent of a PCI/PCI-X host.
		11	MPC8540 acts as the host processor (default).

NOTE

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

4.4.3.5 CPU Boot Configuration

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 18.4.1.2, "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 MPC8540. 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 18.4.1.2, "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 18.4.1.4, "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 18.4.1.4, "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 18.4.1.4, "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 18.4.1.4, "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 MPC8540 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 18.4.1.4, "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 18.4.1.4, "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 18.4.1.3, "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 18.4.1.4, "POR Device Status Register (PORDEVSR)."

Table 4-22. PCI Arbiter Configuration

Functional Signal	Reset Configuration Name	Value (Binary)	Meaning
$\overline{\text{PCI\_GNT2}}$ Default (1)	cfg_pci_arbiter	0	The on-chip PCI/PCI-X arbiter is disabled. External arbitration is required.
		1	The on-chip PCI/PCI-X arbiter is enabled (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 18.4.1.5, "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 18.4.1.4, "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 18.4.1.5, “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 MPC8540 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 18.4.1.5, “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 *MPC8540 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 *MPC8540 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 18.4.1.6, "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 MPC8540 device.

4.4.4.1 System Clock/PCI Clock

The MPC8540 takes a single input clock, SYSCLK, as its primary clock source for the e500 core and all of the devices and interfaces that operate synchronously with the core. 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.

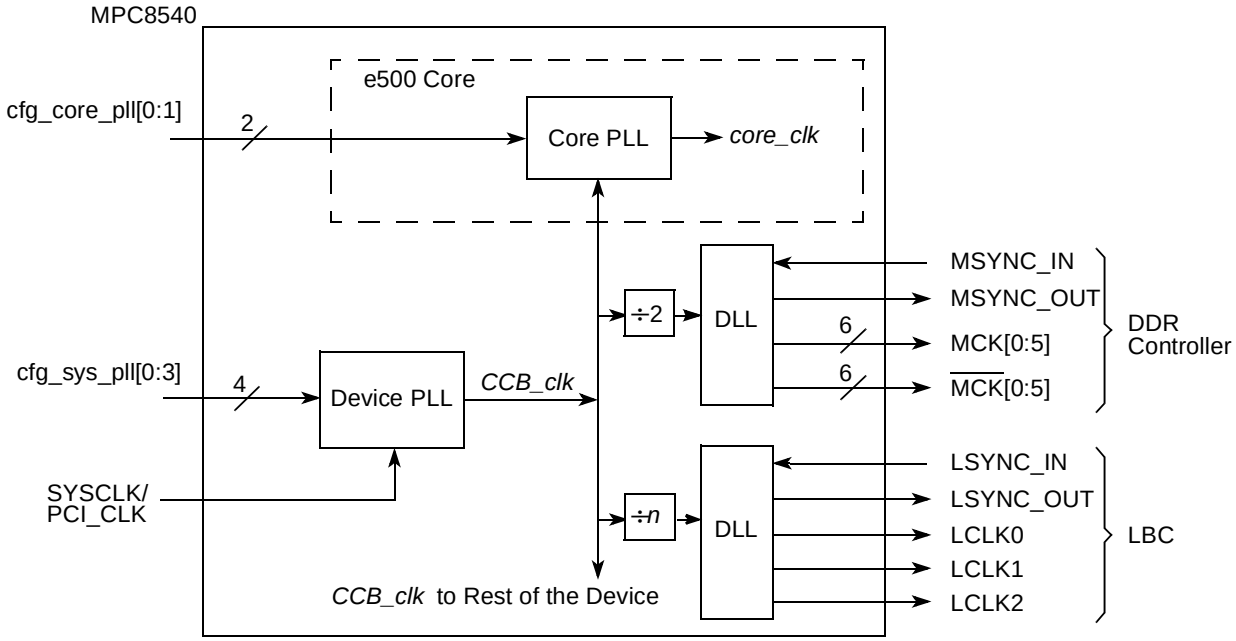


Figure 4-6. Clock Subsystem Block Diagram

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 MPC8540 is in agent mode and host mode. The MPC8540 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 MPC8540 RapidIO port is connected, then the transmit clock can be derived from the receive clock (RIO_RCLK and $\overline{\text{RIO\_RCLK}}$).

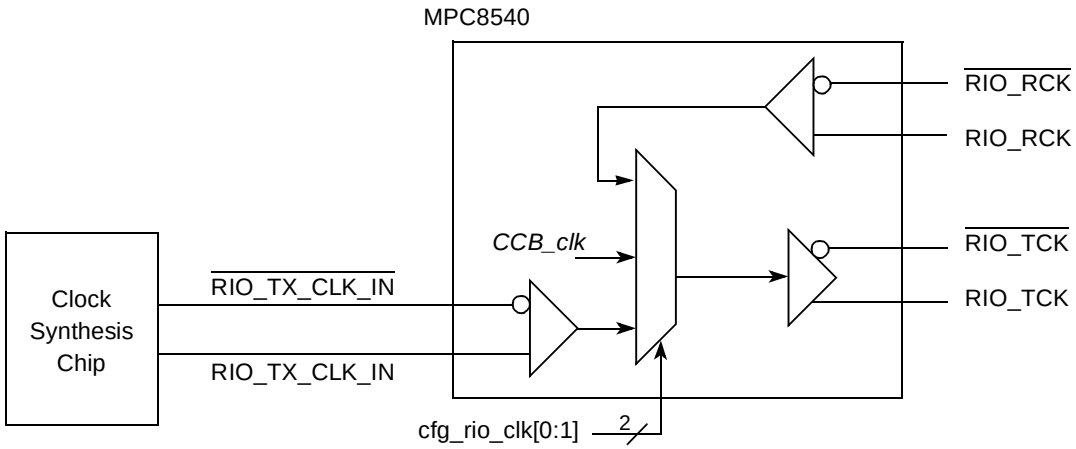


Figure 4-7. RapidIO Transmit Clock Options

If the MPC8540 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 MPC8540 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 *MPC8540 Integrated Processor Hardware Specifications*, but the maximum value should not exceed one-eighth of the core frequency.

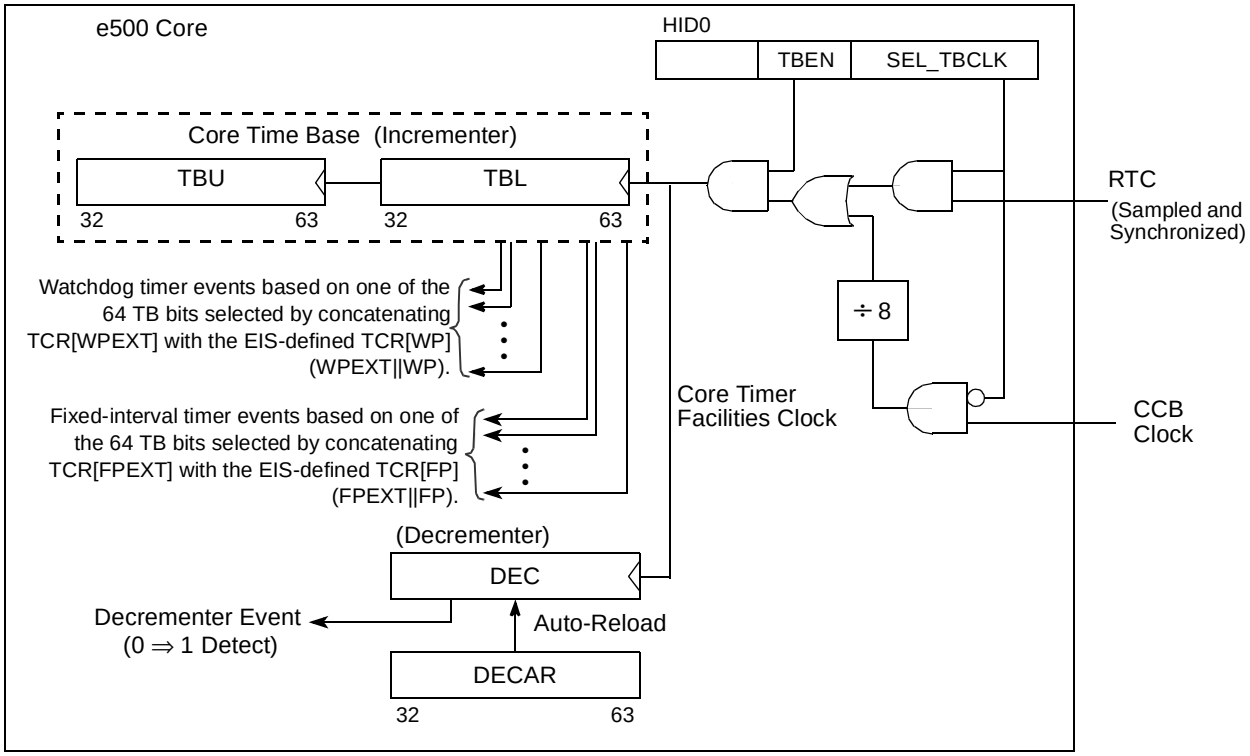
Before being distributed to the core time base, RTC is sampled and synchronized with the CCB clock.

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 MPC8540 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 MPC8540. Note that the L2 cache can also be addressed directly as memory-mapped SRAM.

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

The e500 core complex interacts with the L2 cache 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 MPC8540.

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.

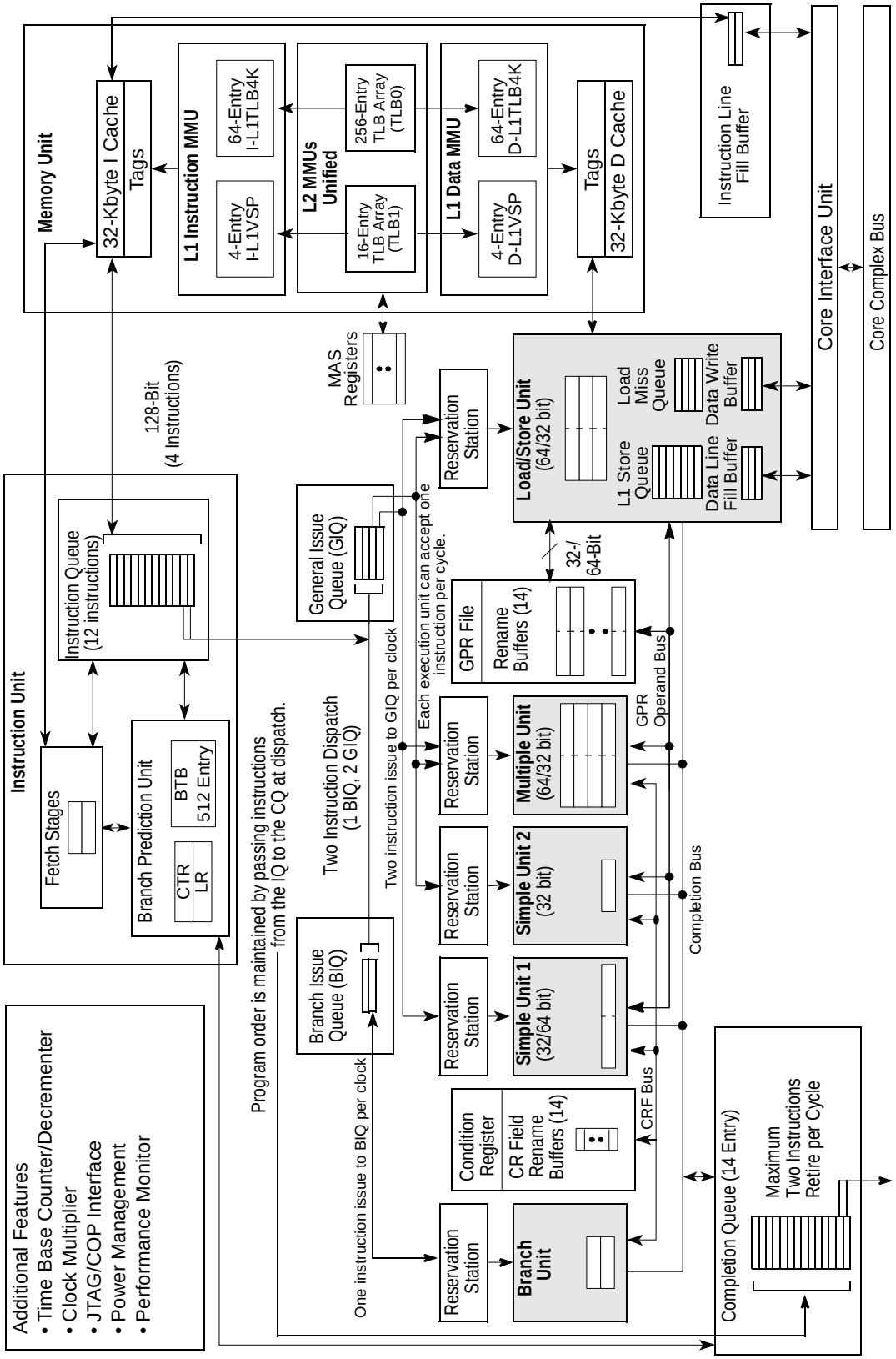


Figure 5-1. e500 Core Complex Block Diagram

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

MPC8540 Revision	e500 Core Revision	Processor Version Register (PVR)	System Version Register (SVR)
1.0	1.0	0x8020_0010	0x8030_0010
2.0	2.0	0x8021_0010	0x8030_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.

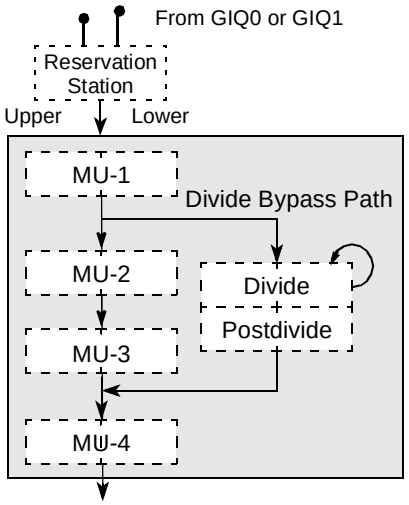


Figure 5-2. Four-Stage MU Pipeline, Showing Divide Bypass

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.

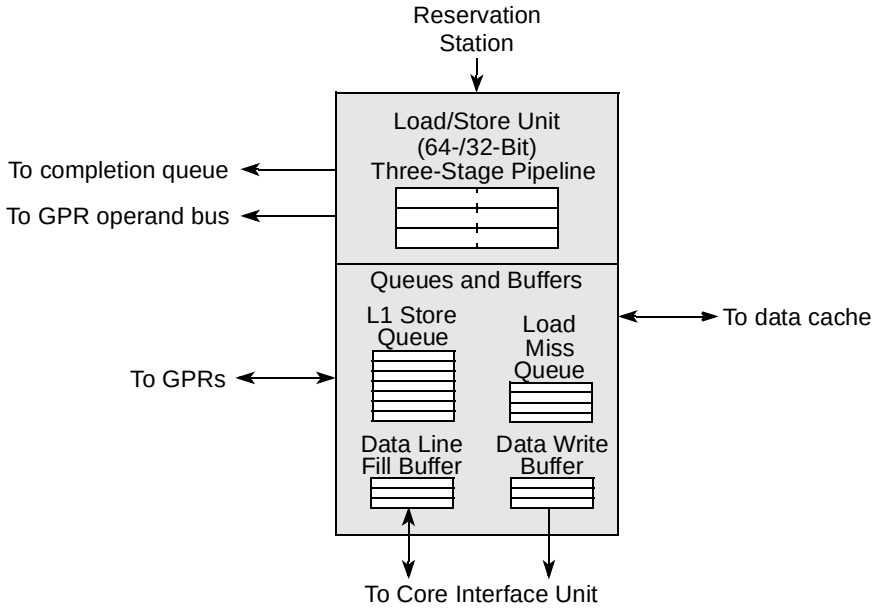


Figure 5-3. Three-Stage Load/Store Unit

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 MPC8540 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
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.

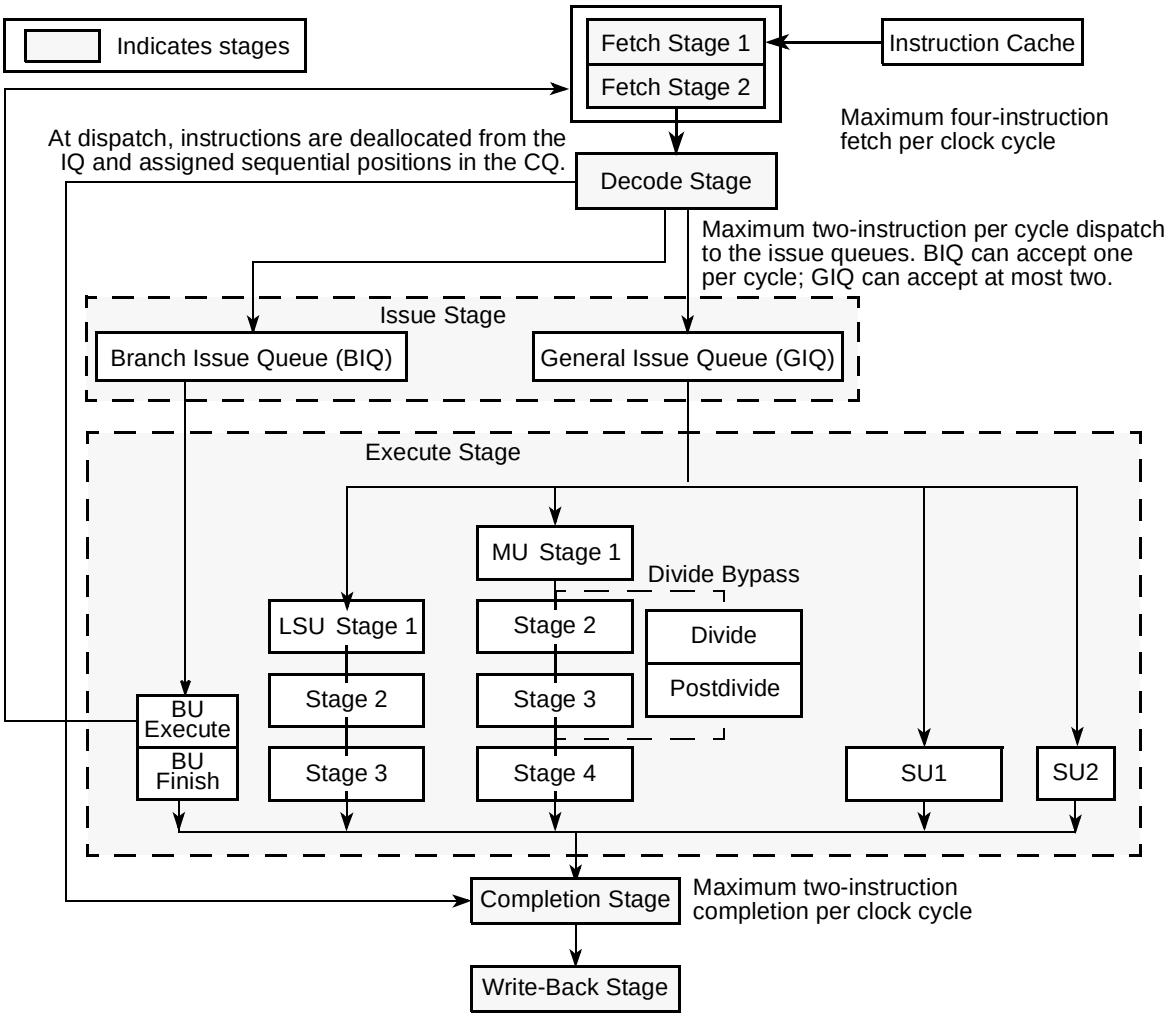


Figure 5-4. Instruction Pipeline Flow

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.

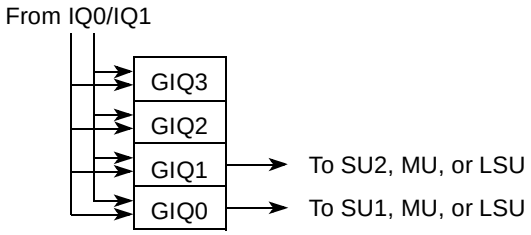


Figure 5-5. GPR Issue Queue (GIQ)

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.

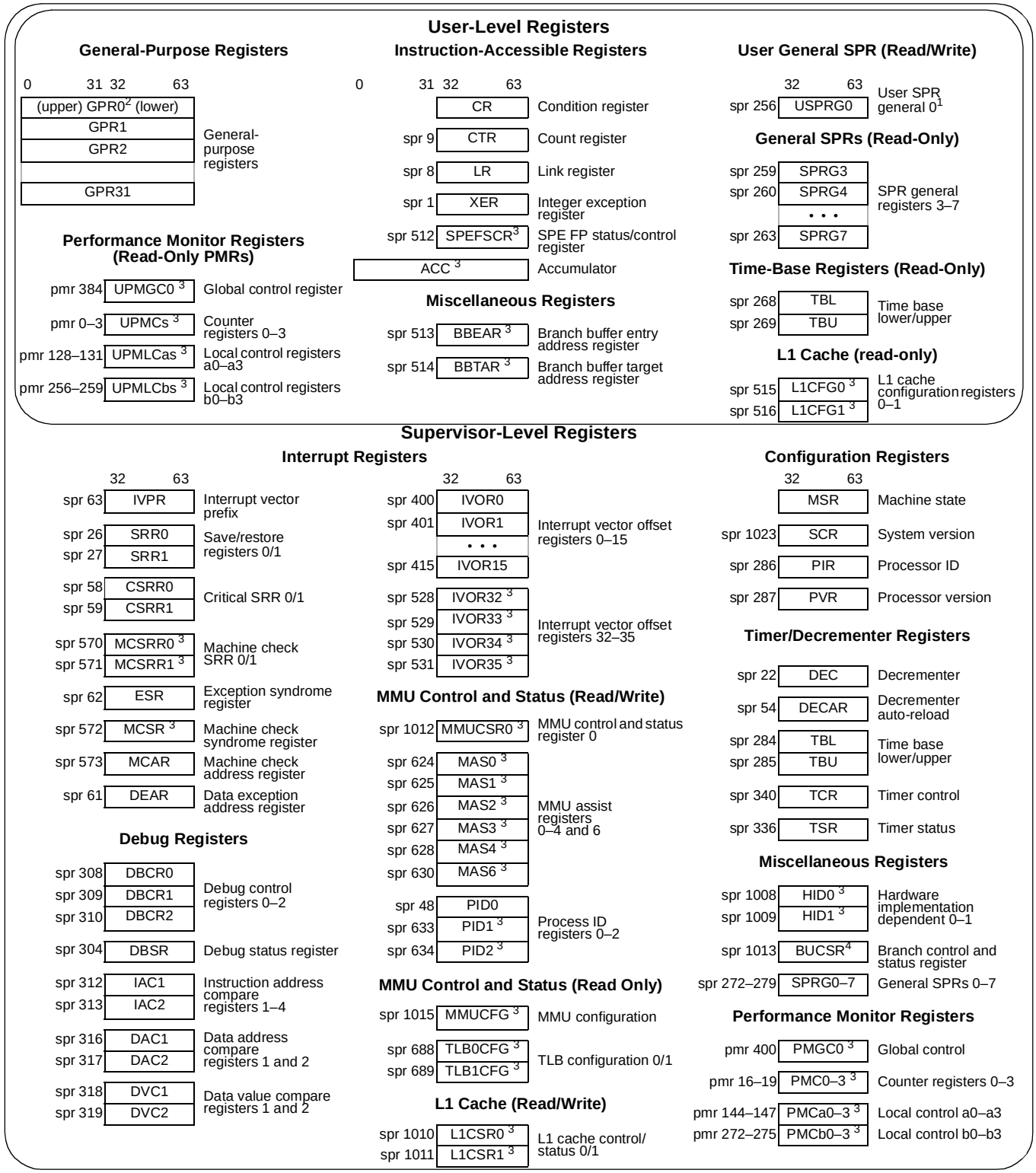
The e500 core also provides new instructions that perform single-instruction, multiple-data (SIMD) operations. These signal processing instructions consist of parallel operations on both the

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 MPC8540 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
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 MPC8540)
- 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.

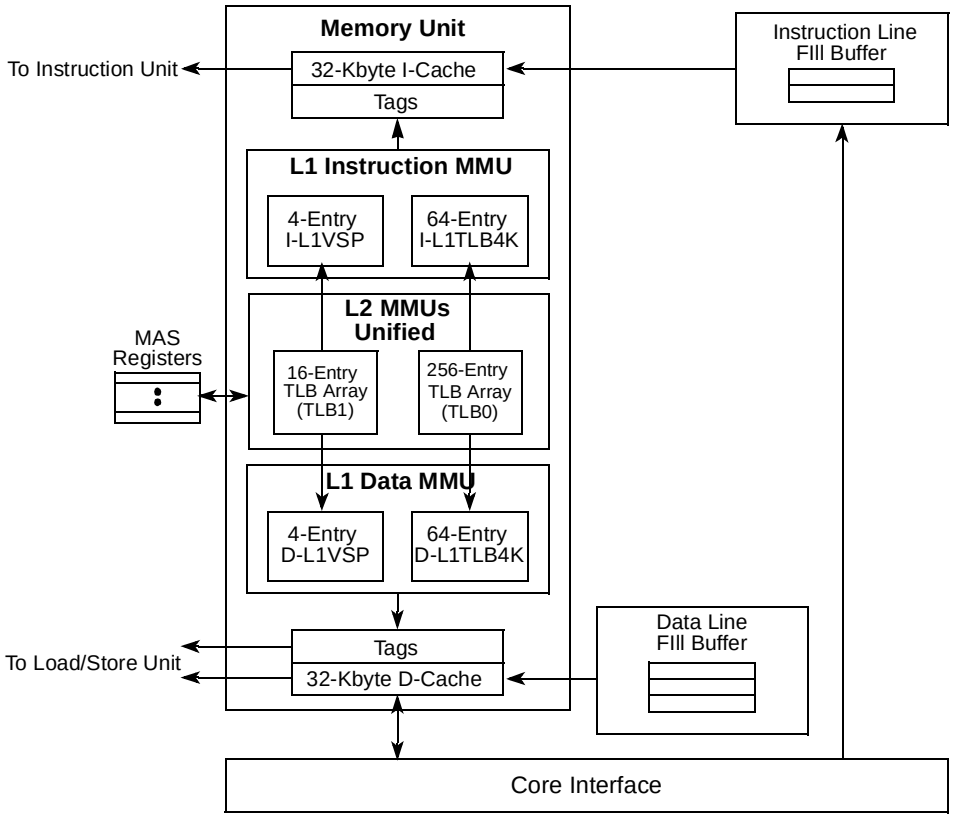


Figure 5-7. MMU Structure

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.

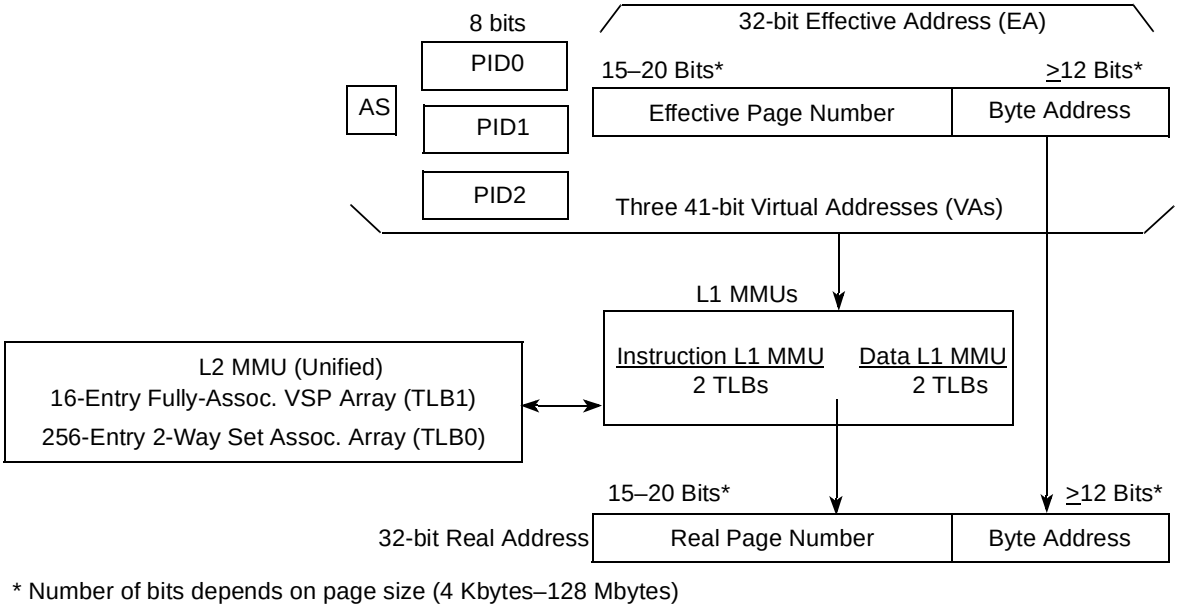


Figure 5-8. Effective-to-Real Address Translation Flow

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 MPC8540 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 16.3.1.4, “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 13.3.1.11, “Transfer Error Check Disable Register (LTEDR),” and Section 13.3.1.12, “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 17.3.2.1.2, “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 is0x8030_*nnnn.* See Table 5-1. SVR[VERSION] = 0x8030 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.

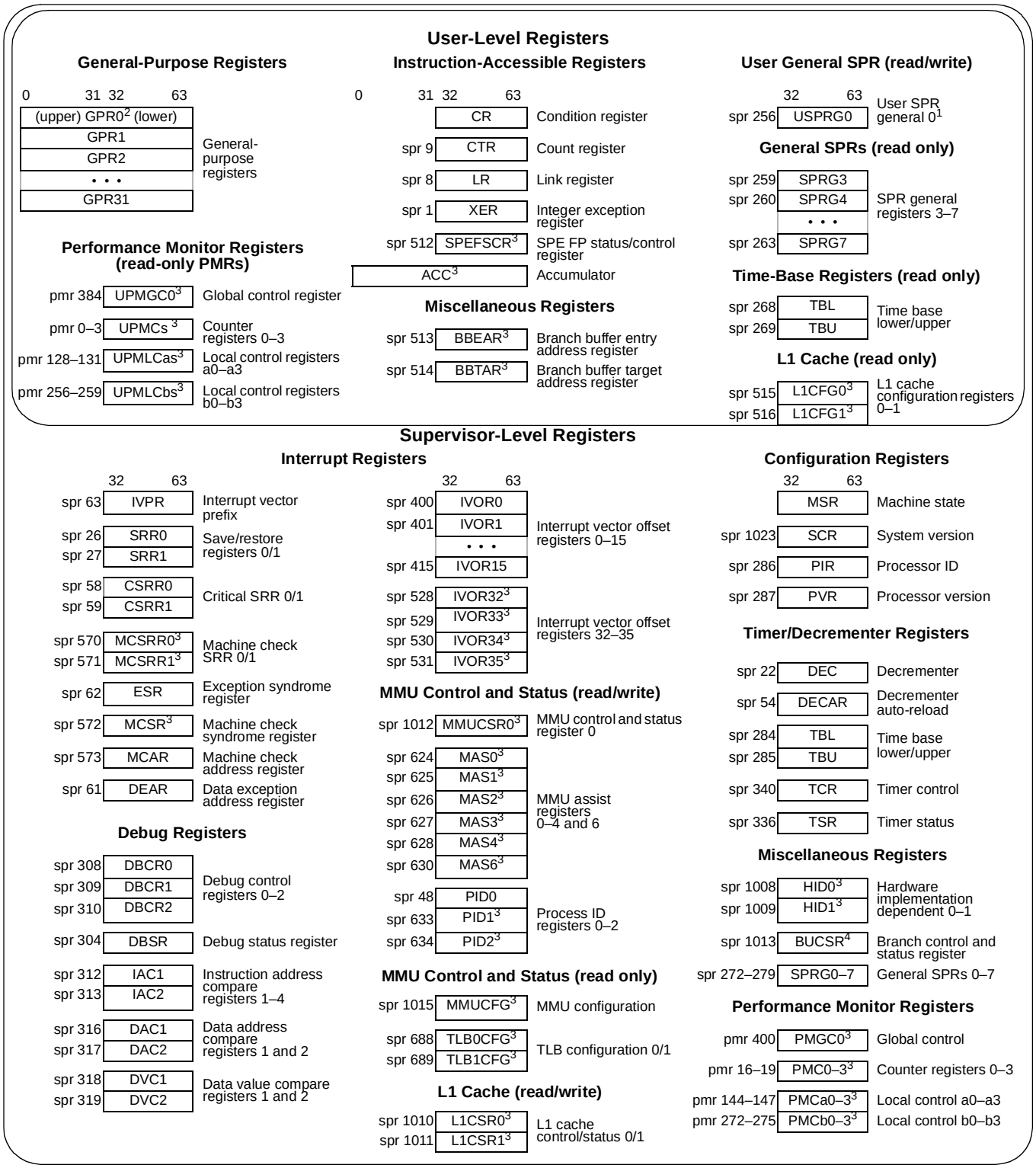
6.1 Overview

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	6.7.1.1/6-18
CSRR1	Critical save/restore register 1	59	00001 11011	Read/Write	Yes	6.7.1.2/6-18
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	6.13.4/6-48
DAC2	Data address compare 2	317	01001 11101			
DBCR0	Debug control register 0	308	01001 10100	Read/Write	Yes	6.13.1/6-42
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	6.13.2/6-46
DEAR	Data exception address register	61	00001 11101	Read/Write	Yes	6.7.1.5/6-19
DEC	Decrementer	22	00000 10110	Read/Write	Yes	6.6.4/6-17
DECAR	Decrementer auto-reload	54	00001 10110	Write-only		
ESR	Exception syndrome register	62	00001 11110	Read/Write	Yes	6.7.1.8/6-20

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	6.13.3/6-47
IAC2	Instruction address compare 2	313	01001 11001			
IVOR0	Critical input	400	01100 10000	Read/Write	Yes	6.7.1.7/6-19
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	6.7.1.6/6-19
LR	Link register	8	00000 01000	Read/Write	No	6.4.2/6-11
PID	Process ID register [2]	48	00001 10000	Read/Write	Yes	6.12.1/6-34
PIR	Processor ID register	286	01000 11110	Read-only	Yes	6.5.2/6-14
PVR	Processor version register	287	01000 11111	Read-only	Yes	6.5.3/6-14
SPRG0	SPR general 0	272	01000 10000	Read/Write	Yes	6.8/6-24
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	

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			
SPRG5	SPR general 5	261	01000 00101	Read-only	No	6.8/6-24
		277	01000 10101	Read/Write	Yes	
SPRG6	SPR general 6	262	01000 00110	Read-only	No	
		278	01000 10110	Read/Write	Yes	
SPRG7	SPR general 7	263	01000 00111	Read-only	No	
		279	01000 10111	Read/Write	Yes	
SRR0	Save/restore register 0	26	00000 11010	Read/Write	Yes	6.7.1.1/6-18
SRR1	Save/restore register 1	27	00000 11011	Read/Write	Yes	6.7.1.2/6-18
TBL	Time base lower	268	01000 01100	Read-only	No	6.6.3/6-17
		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	6.6.1/6-15
TSR	Timer status register	336	01010 10000	Read/Clear[4]	Yes	6.6.2/6-16
USPRG0	User SPR general 0[5]	256	01000 00000	Read/Write	No	6.8/6-24
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	6.9.1/6-25
BBTAR	Branch buffer target address register	514	Read/Write	No	6.9.2/6-25
BUCSR	Branch unit control and status register	1013	Read/Write	Yes	6.9.3/6-26
HID0	Hardware implementation dependent reg 0	1008	Read/Write	Yes	6.10.1/6-27
HID1	Hardware implementation dependent reg 1	1009	Read/Write	Yes	6.10.1/6-27

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

SPR Abbreviation	Name	SPR Number	Access	Supervisor Only	Section/ Page
IVOR32	SPE APU unavailable interrupt offset	528	Read/Write	Yes	6.7.1.7/6-19
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	6.11.3/6-32
L1CFG1	L1 cache configuration register 1	516	Read-only	No	6.11.4/6-33
L1CSR0	L1 cache control and status register 0	1010	Read/Write	Yes	6.11.1/6-30
L1CSR1	L1 cache control and status register 1	1011	Read/Write	Yes	6.11.2/6-31
MAS0	MMU assist register 0	624	Read/Write	Yes	6.12.5.1/6-37
MAS1	MMU assist register 1	625	Read/Write	Yes	6.12.5.2/6-38
MAS2	MMU assist register 2	626	Read/Write	Yes	6.12.5.3/6-39
MAS3	MMU assist register 3	627	Read/Write	Yes	6.12.5.4/6-40
MAS4	MMU assist register 4	628	Read/Write	Yes	6.12.5.5/6-40
MAS6	MMU assist register 6	630	Read/Write	Yes	6.12.5.6/6-41
MCAR	Machine check address register	573	Read-only	Yes	6.7.2.3/6-22
MCSR	Machine check syndrome register	572	Read/Write	Yes	6.7.2.4/6-23
MCSRR0	Machine check save/restore register 0	570	Read/Write	Yes	6.7.2.1/6-21
MCSRR1	Machine check save/restore register 1	571	Read/Write	Yes	6.7.2.2/6-22
MMUCFG	MMU configuration register	1015	Read-only	Yes	6.12.3/6-35
MMUCSR0	MMU control and status register 0	1012	Read/Write	Yes	6.12.2/6-34
PID0	Process ID register 0. Book E refers to this as PID instead of PID0.	48	Read/Write	Yes	6.12.1/6-34
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	6.14/6-48
SVR	System version register	1023	Read-only	Yes	6.5.4/6-14

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

SPR Abbreviation	Name	SPR Number	Access	Supervisor Only	Section/ Page
TLB0CFG	TLB configuration register 0	688	Read-only	Yes	6.12.4/6-35
TLB1CFG	TLB configuration register 1	689	Read-only	Yes	6.12.4.2/6-36

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)

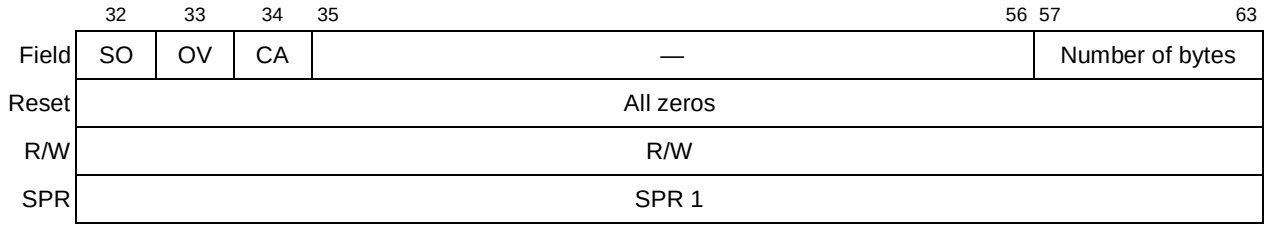


Figure 6-2. Integer Exception Register (XER)

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)

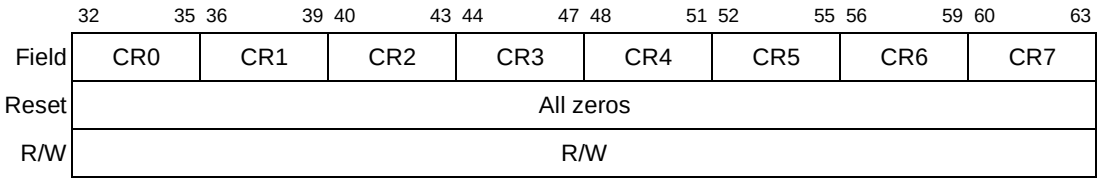


Figure 6-3. Condition Register (CR)

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.

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

CR*n* Bits	CR Bits	BI	Description
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.
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)

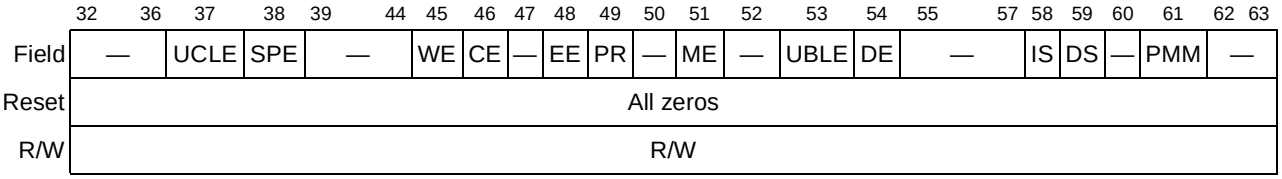


Figure 6-6. Machine State Register (MSR)

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.
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

Table 6-6. MSR Field Descriptions (continued)

Bits	Name	Description
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).
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)

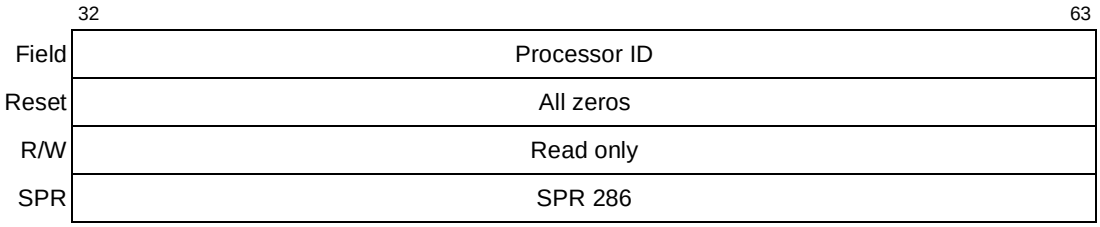


Figure 6-7. Processor ID Register (PIR)

6.5.3 Processor Version Register (PVR)

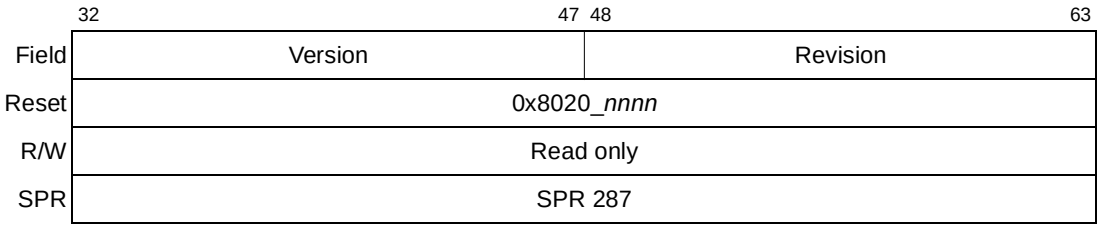


Figure 6-8. Processor Version Register (PVR)

Table 6-7. PVR Field Descriptions

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

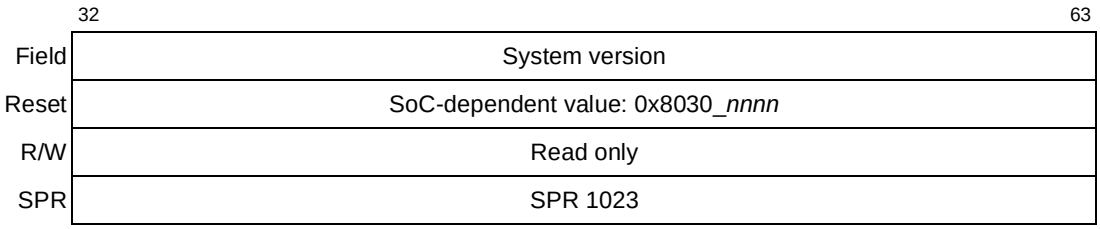


Figure 6-9. System Version Register (SVR)

6.6 Timer Registers

6.6.1 Timer Control Register (TCR)

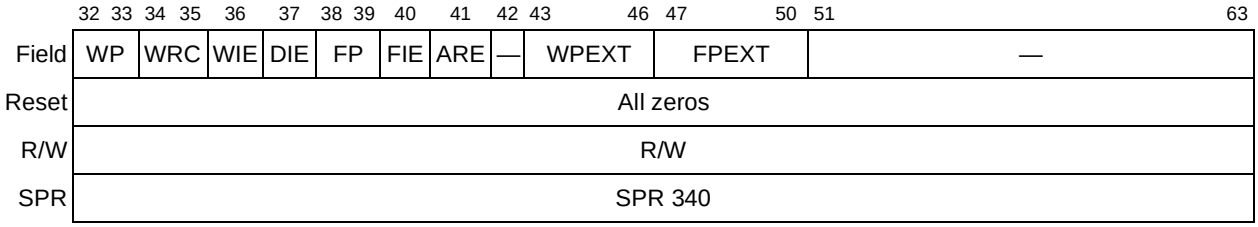


Figure 6-10. Timer Control Register (TCR)

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
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)

Table 6-8. TCR Field Descriptions (continued)

Bits	Name	Description
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.
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)

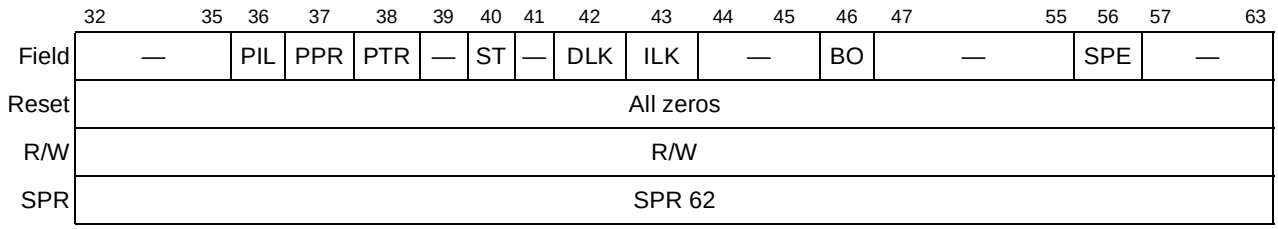


Figure 6-22. Exception Syndrome Register (ESR)

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.	

Table 6-12. MCSR Field Descriptions (continued)

Bit	Name	Description	Recoverable
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)

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

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

Table 6-14. BBTAR Field Descriptions (continued)

Bits	Name	Description
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)

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

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.
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
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

Table 6-16. HID0 Field Descriptions (continued)

Bits	Name	Description
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 MPC8540
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. MPC8540 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 MPC8540**,** 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)

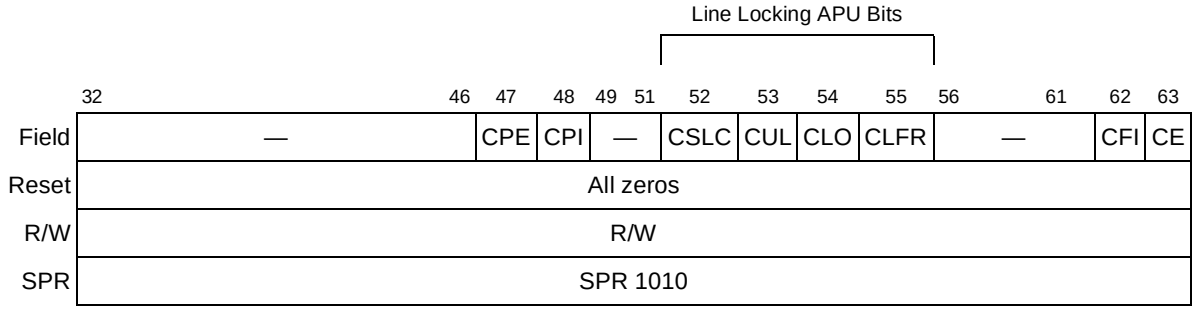


Figure 6-33. L1 Cache Control and Status Register 0 (L1CSR0)

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)

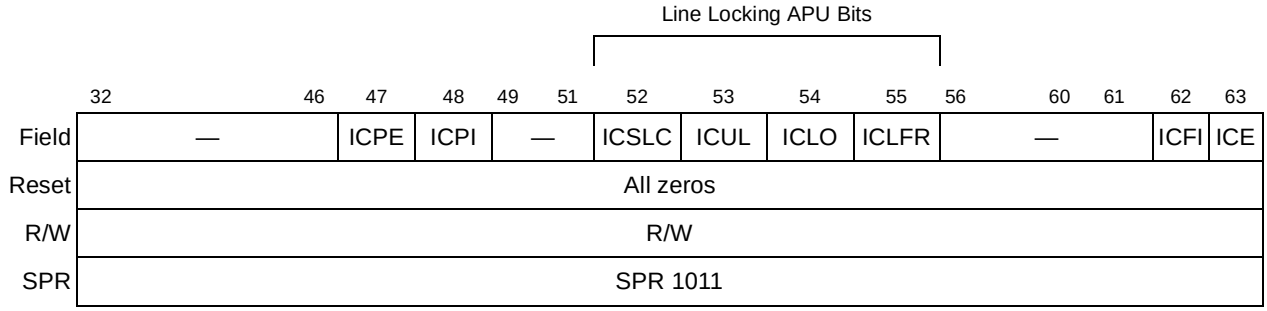


Figure 6-34. L1 Cache Control and Status Register 1 (L1CSR1)

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

Table 6-20. L1CFG0 Field Descriptions (continued)

Bits	Name	Description
44	CPA	Cache parity available 0 Unavailable 1 Available
45–49	—	Reserved, should be cleared.
50–52	CNWAY	Cache number of ways 111 Indicates 8 ways
53–55	—	Reserved, should be cleared.
56–63	CSIZE	Cache size 0x20 indicates 32 Kbytes

6.11.4 L1 Cache Configuration Register 1 (L1CFG1)

	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)

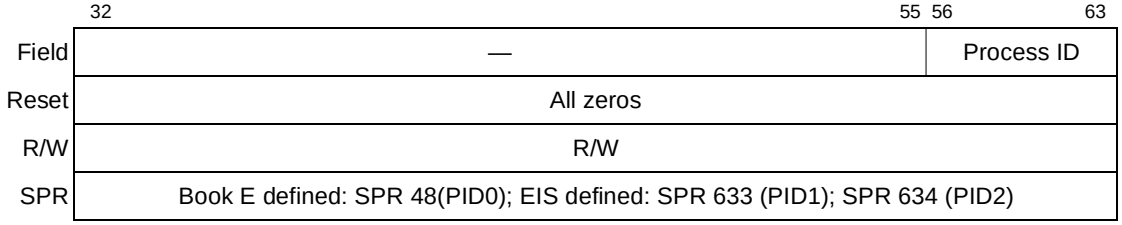


Figure 6-37. Process ID Registers (PID0–PID2)

6.12.2 MMU Control and Status Register 0 (MMUCSR0)

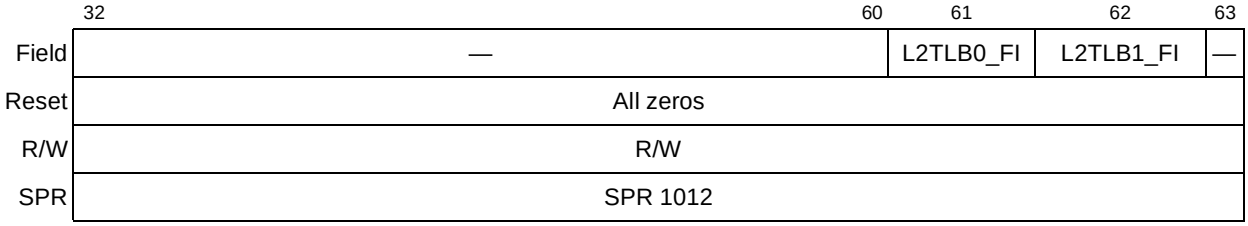


Figure 6-38. MMU Control and Status Register 0 (MMUCSR0)

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
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

Table 6-25. TLB1CFG Field Descriptions (continued)

Bits	Name	Description
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)

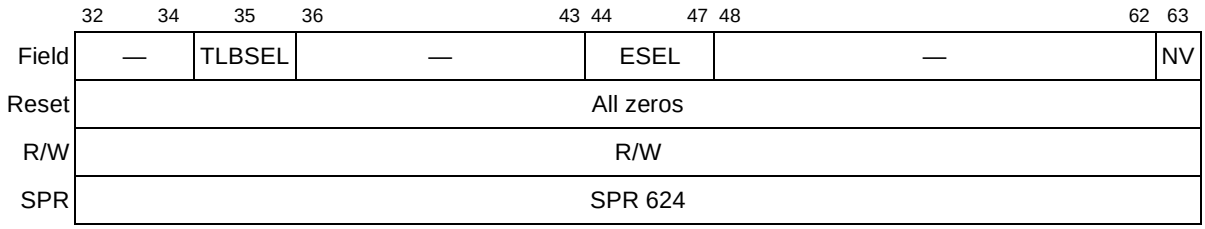


Figure 6-42. MAS Register 0 (MAS0)

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)

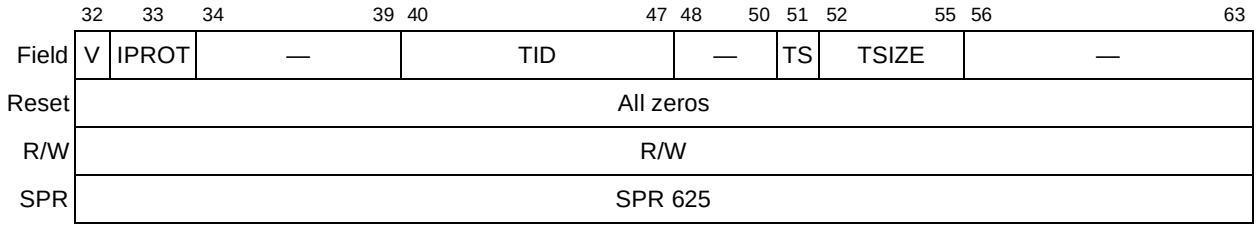


Figure 6-43. MAS Register 1 (MAS1)

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)

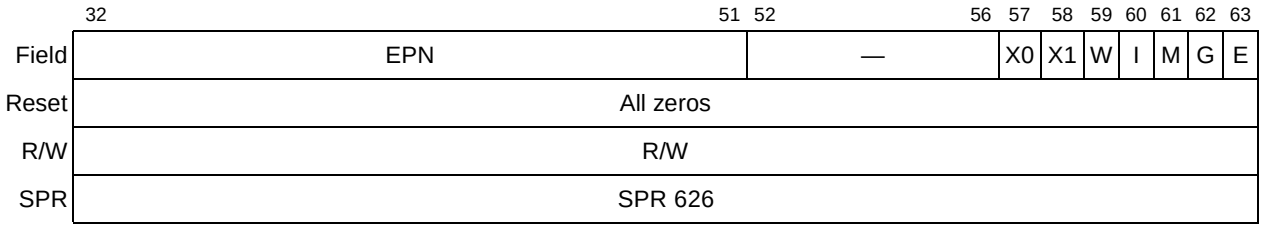


Figure 6-44. MAS Register 2 (MAS2)

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.

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

Bits	Name	Description
46–47	TIDSELD	TID default selection value. A 2-bit field that specifies which of the current PID registers should be used to load the MAS1[TID] field on a TLB miss exception.
		The e500 implementation defines this field as follows: 00 PID0 01 PID1 10 PID2 11 TIDZ (0x00) (all zeros)
48–51	—	Reserved, should be cleared.
52–55	TSIZED	Default TSIZE value. Specifies the default value to be loaded into MAS1[TSIZE] on a TLB miss exception.
56	—	Reserved, should be cleared.
57	X0D	Default X0 value. Specifies the default value to be loaded into MAS2[X0] on a TLB miss exception.
58	X1D	Default X1 value. Specifies the default value to be loaded into MAS2[X1] on a TLB miss exception.
59	WD	Default W value. Specifies the default value to be loaded into MAS2[W] on a TLB miss exception.
60	ID	Default I value. Specifies the default value to be loaded into MAS2[I] on a TLB miss exception.
61	MD	Default M value. Specifies the default value to be loaded into MAS2[M] on a TLB miss exception.
62	GD	Default G value. Specifies the default value to be loaded into MAS2[G] on a TLB miss exception.
63	ED	Default E value. Specifies the default value to be loaded into MAS2[E] on a TLB miss exception.

6.12.5.6 MAS Register 6 (MAS6)

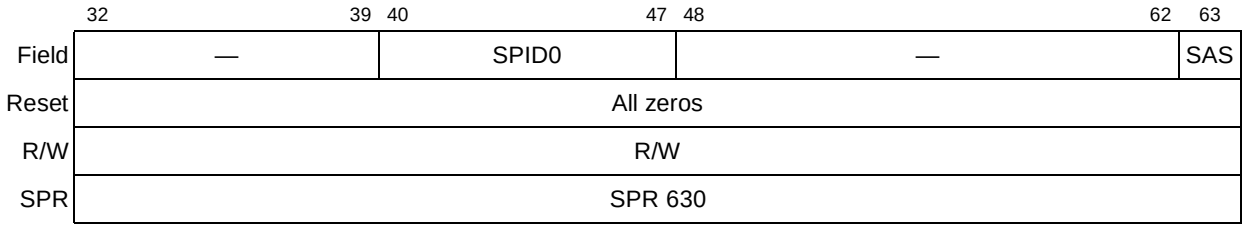


Figure 6-47. MAS Register 6 (MAS6)

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)

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

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

Table 6-32. DBCR0 Field Descriptions (continued)

Bits	Name	Description
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
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
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)

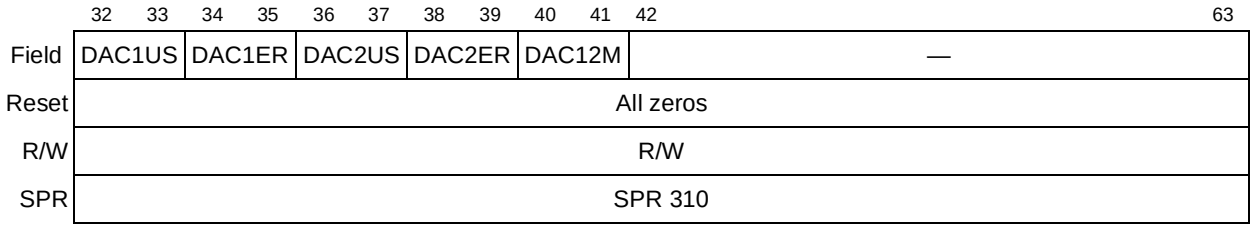


Figure 6-50. Debug Control Register 2 (DBCR2)

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.
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.

Table 6-34. DBCR2 Field Descriptions (continued)

Bits	Name	Description
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 UDE (level sensitive, active low) is asserted, DBSR[UDE] is affected as follows: MSR[DE]DBCR0[IDM]Action X 0 No action. 0 1 UDE is set. 1 1 UDE is set and a debug interrupt is taken.
34–35	MRR	Most recent reset. Functions as write-one-to-clear. Undefined at power-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.

Table 6-35. DBSR Field Descriptions (continued)

Bits	Name	Description
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)

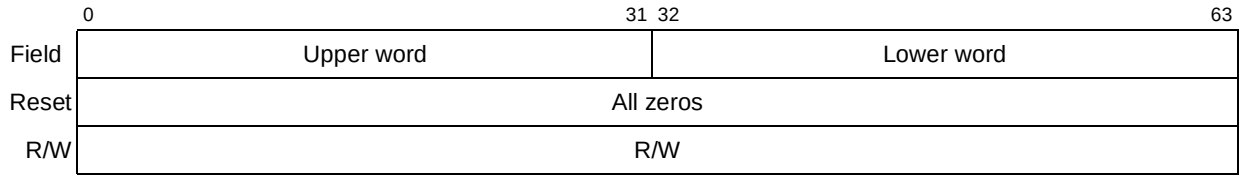


Figure 6-55. Accumulator (ACC)

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	6.15.4/6-54
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	6.15.1/6-52
PMLCa0	Performance monitor local control a0	144	00100	10000	6.15.2/6-52
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	6.15.3/6-53
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	6.15.4/6-54
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	6.15.3/6-53
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	6.15.3/6-53
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	6.15.2/6-52

6.15.1 Global Control Register 0 (PMGC0, UPMGC0)

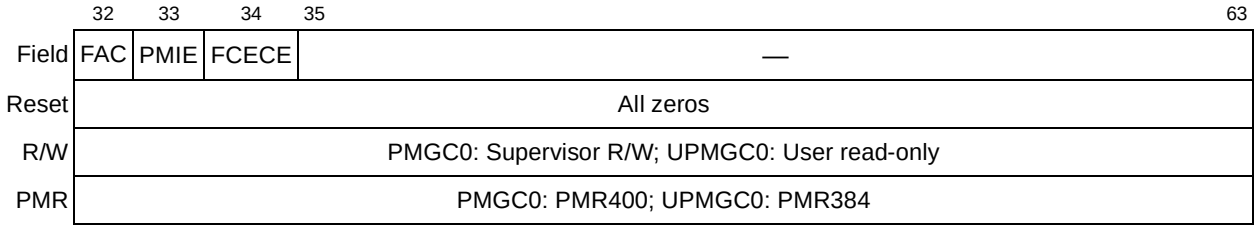


Figure 6-56. Performance Monitor Global Control Register 0 (PMGC0), User Performance Monitor Global Control Register 0 (UPMGC0)

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)

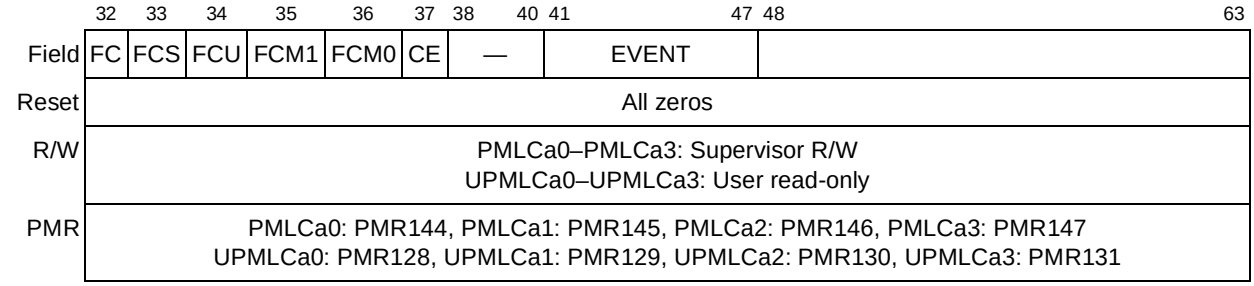


Figure 6-57. Local Control A Registers (PMLCa0–PMLCa3), User Local Control A Registers (UPMLCa0–UPMLCa3)

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.

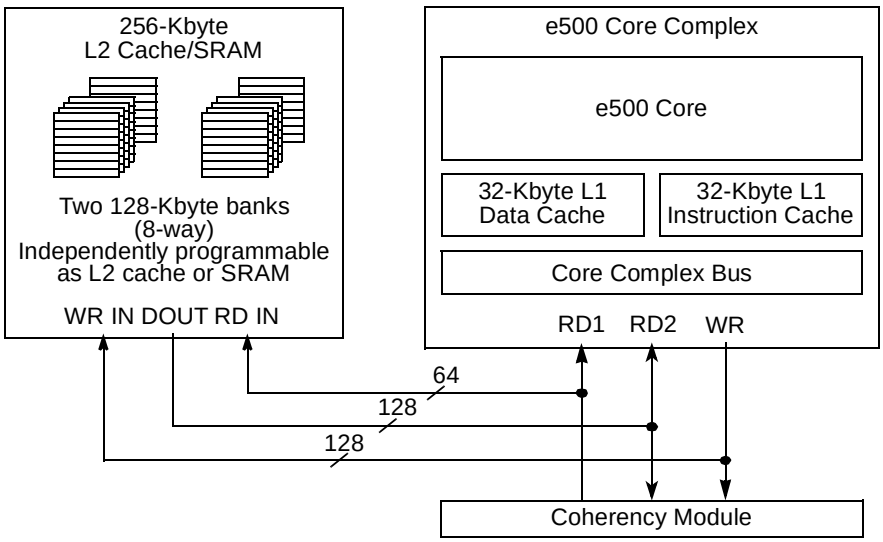


Figure 7-1. L2 Cache/SRAM Configuration

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 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.

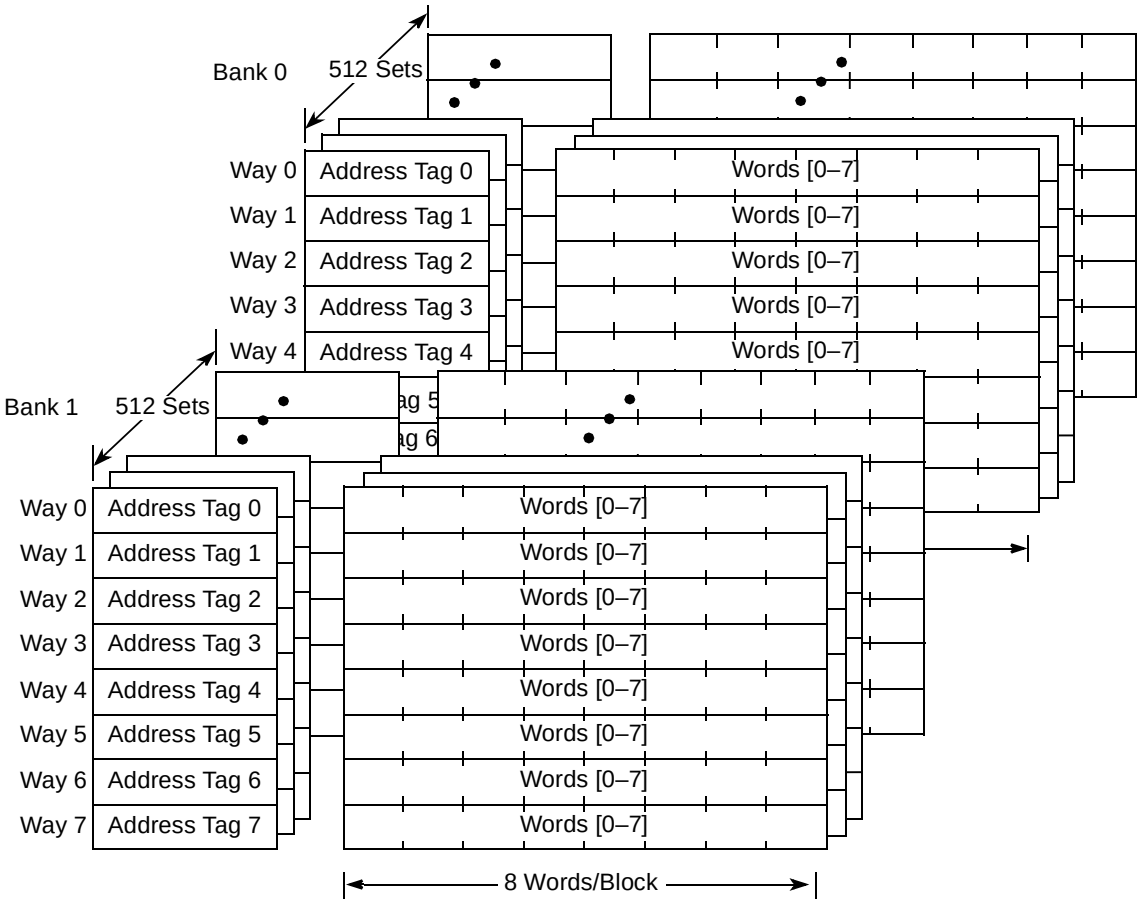


Figure 7-2. Cache Organization

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).

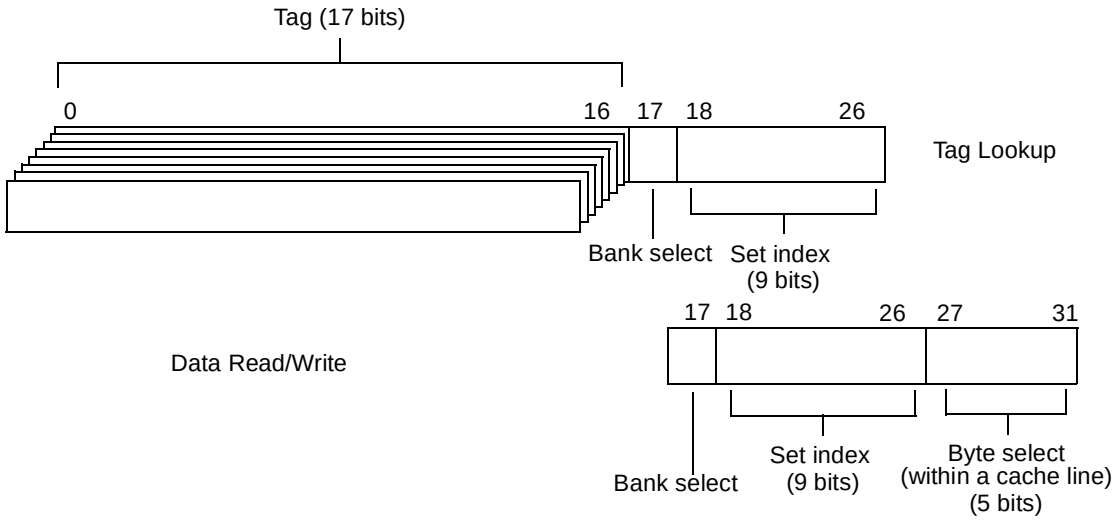


Figure 7-3. 256-Kbyte L2 Cache Address Configuration—Full Cache Mode

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.

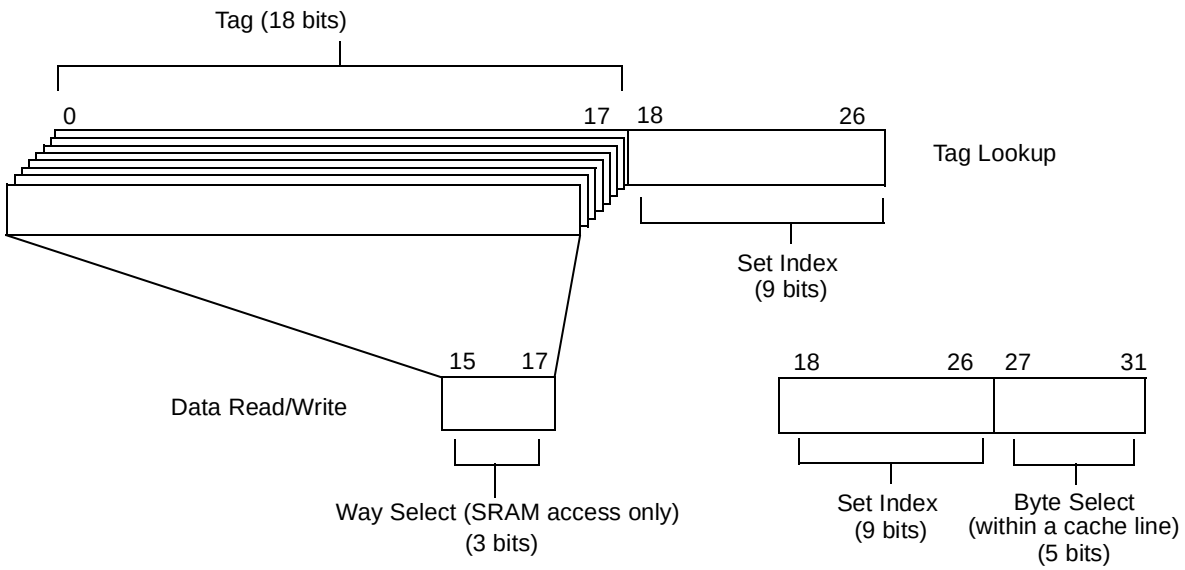


Figure 7-4. 128-Kbyte L2 Cache Address Configuration—Half SRAM, Half Cache Mode

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.

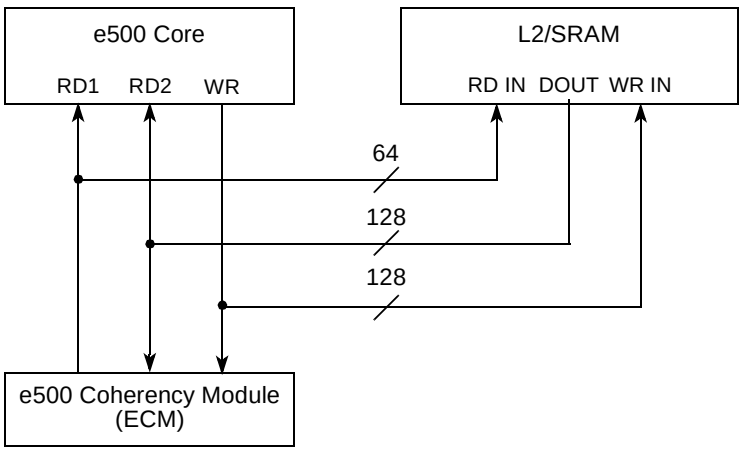


Figure 7-5. Data Bus Connection of CCB

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

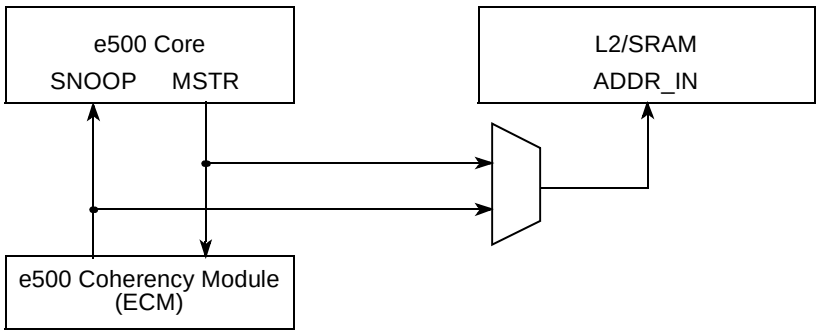


Figure 7-6. Address Bus Connection of CCB

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	7.3.1.3/7-10
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	7.3.1.3/7-10
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	7.3.1.3/7-10
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	7.3.1.3/7-10
0x2_0100	L2SRBAR0—L2 memory-mapped SRAM base address register 0	R/W	0x0000_0000	7.3.1.4/7-11
0x2_0108	L2SRBAR1—L2 memory-mapped SRAM base address register 1	R/W	0x0000_0000	7.3.1.4/7-11
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

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

Offset	Register	Access	Reset	Section/Page
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**

	0	1	2	3	4	5	6		8	9	10	11	12	13	15
R	L2E	L2I	L2SIZ		L2BLKSZ		0	0	0	L2DO	L2IO	0	L2INTDIS	L2SRAM	
W															
Reset	0010_0000_0000_0000														

	16	17	18	19	20	21	22	23	24						31
R	0	0	L2LO	L2SLC	0	L2LFR	L2LFRID		0	0	0	0	0	0	0
W															
Reset	0000_0000_0000_0000														
Offset	0x2_0000														

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
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

Table 7-2. L2CTL Field Descriptions (continued)

Bits	Name	Description
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.
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 MPC8540 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.

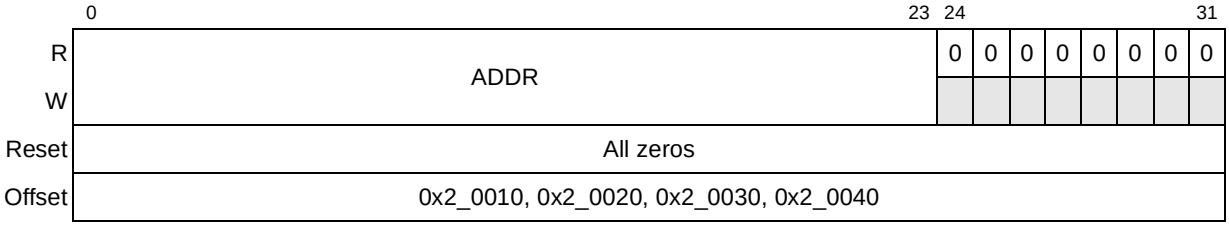


Figure 7-8. L2 Cache External Write Address Registers (L2CEWAR*n*)

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.

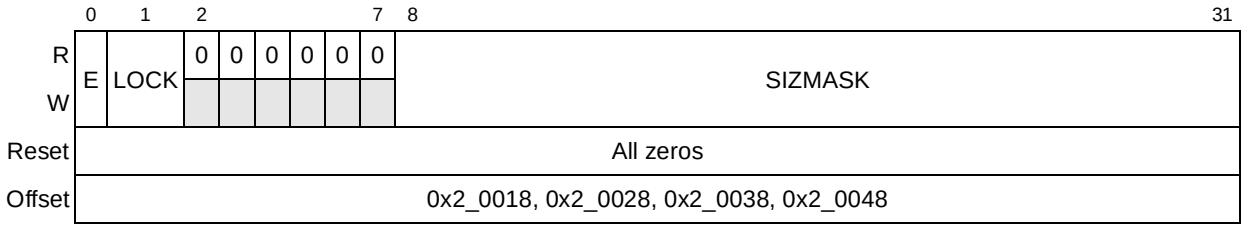


Figure 7-9. L2 Cache External Write Control Registers (L2CEWCR0–L2CEWCR3)

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
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.

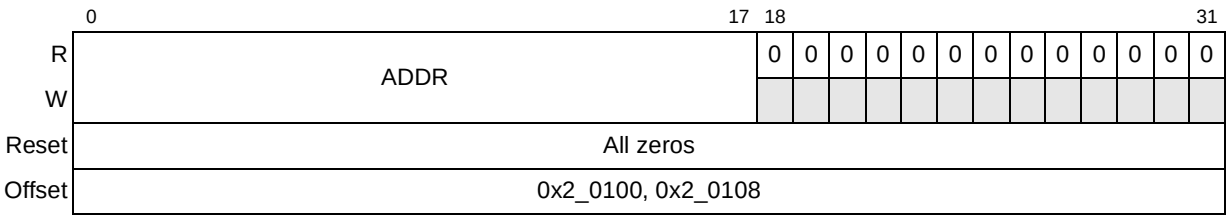


Figure 7-10. L2 Memory-Mapped SRAM Base Address Registers (L2SRBAR*n*)

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 4.3.1.1.2, "Configuration, Control, and Status Base Address Register (CCSRBAR)"). Overlapping SRAM and local access windows is discouraged because processor and snoopable I/O transactions would map to the SRAM while non-snooped I/O transactions would be mapped by the local access windows. Only if all accesses to the SRAM address range are snoopable can results be consistent if SRAM and local access windows overlap. 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 MPC8540 detects an L2 error, the appropriate bit in the error detect register (L2ERRDET) is set. Error detection is disabled by setting the corresponding bit in the error disable register (L2ERRDIS).

The address and attributes of the first detected error are also saved in the error capture registers (L2ERRADDR, L2ERRATTR, L2CAPTDATAHI, L2CAPTDATALO, and L2CAPTACC). Subsequent errors set error bits in the error detection registers, but 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).

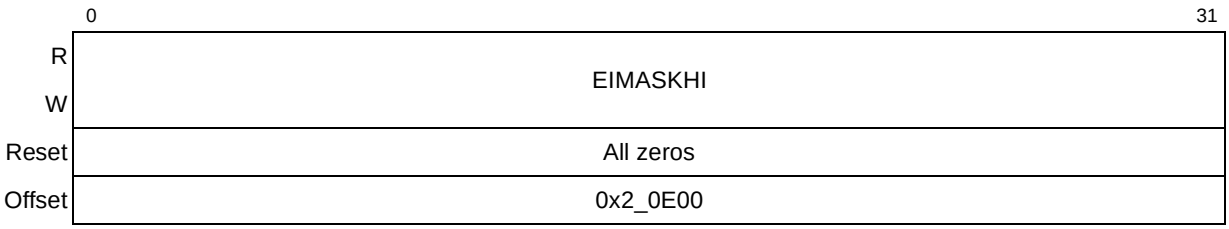


Figure 7-11. L2 Error Injection Mask High Register (L2ERRINJHI)

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.

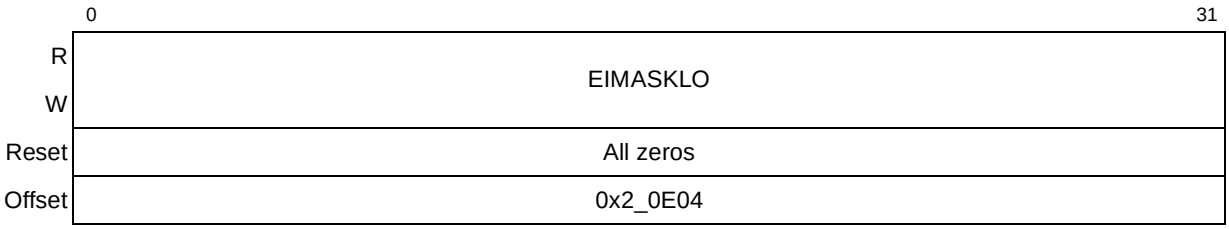


Figure 7-12. L2 Error Injection Mask Low Register (L2ERRINJLO)

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).

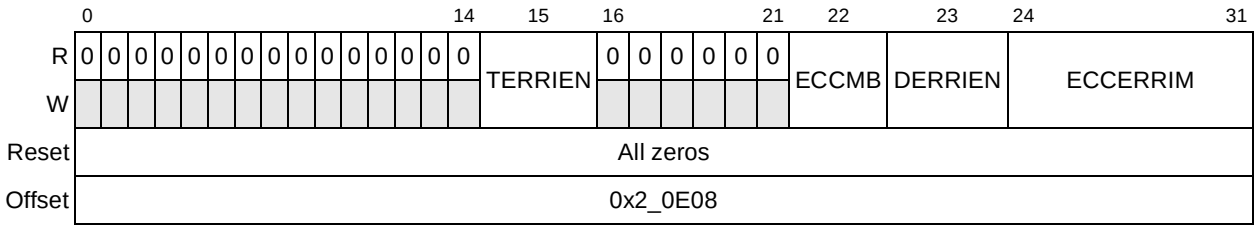


Figure 7-13. L2 Error Injection Mask Control Register (L2ERRINJCTL)

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.

Table 7-8. L2ERRINJCTL Field Descriptions (continued)

Bits	Name	Description
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).

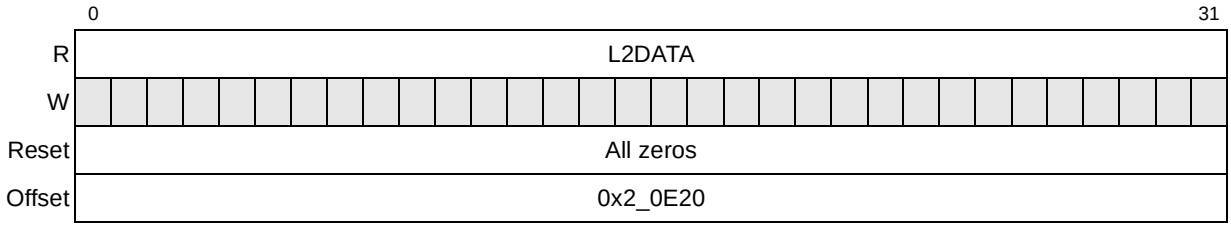


Figure 7-14. L2 Error Capture Data High Register (L2CAPTDATAHI)

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).

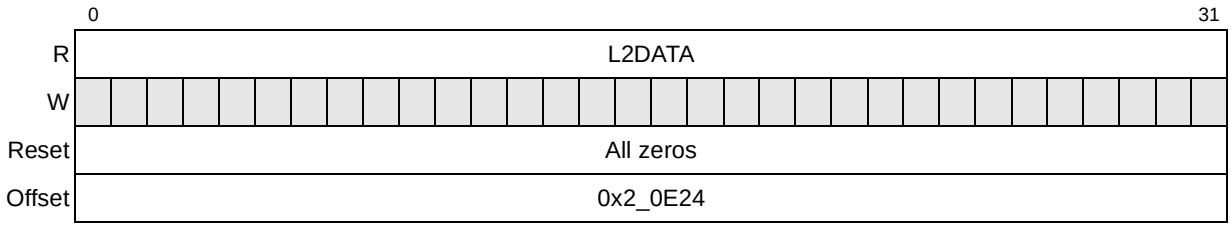


Figure 7-15. L2 Error Capture Data Low Register (L2CAPTDATALO)

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).

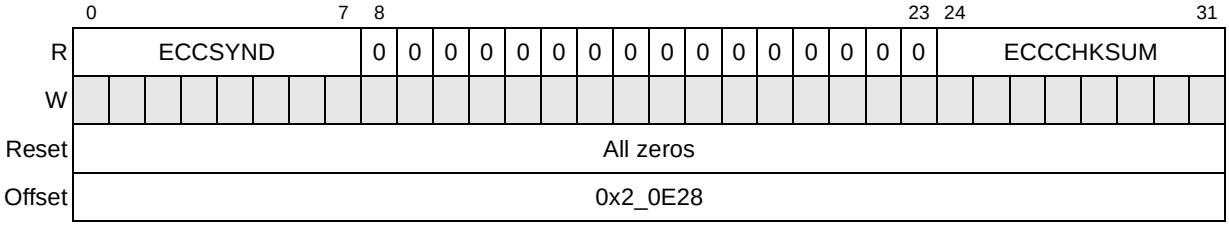


Figure 7-16. L2 Error Syndrome Register (L2CAPTECC)

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).

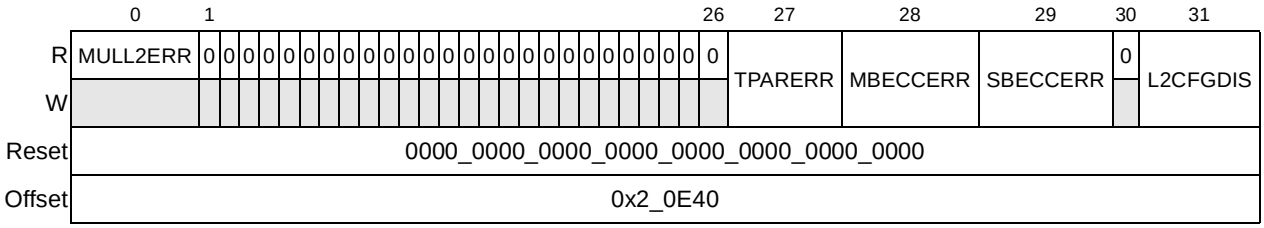


Figure 7-17. L2 Error Detect Register (L2ERRDET)

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

Table 7-12. L2ERRDET Field Descriptions (continued)

Bits	Name	Description
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).

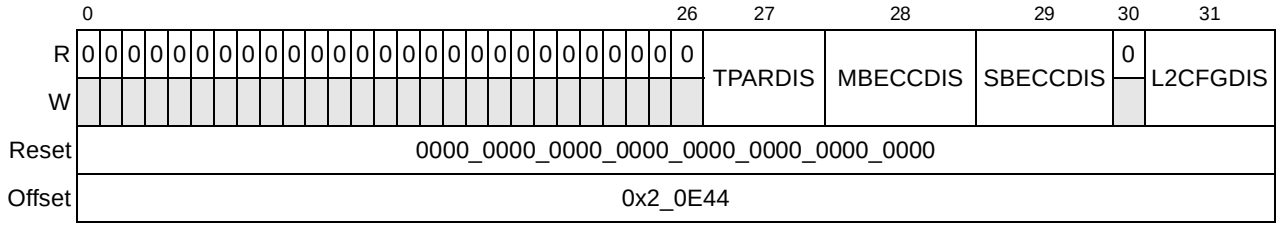


Figure 7-18. L2 Error Disable Register (L2ERRDIS)

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

Table 7-13. L2ERRDIS Field Descriptions (continued)

Bits	Name	Description
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.

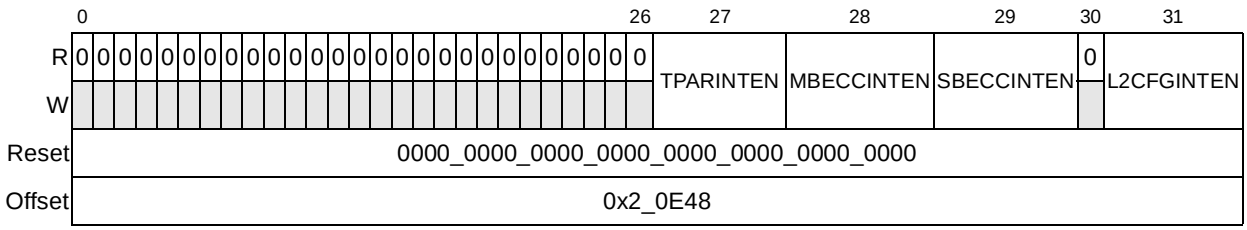


Figure 7-19. L2 Error Interrupt Enable Register (L2ERRINTEN)

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).

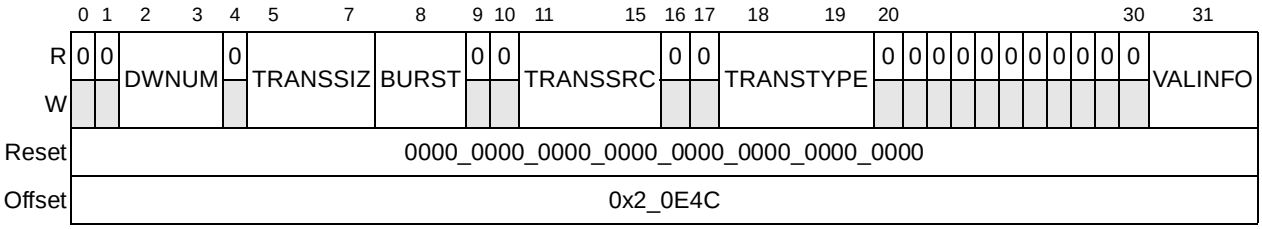


Figure 7-20. L2 Error Attributes Capture Register (L2ERRATTR)

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
5–7	TRANSSIZ	Transaction size for detected error Single-beat / Burst 000 8 bytes / Reserved 001 1 byte / 16 bytes 010 2 bytes / 32 bytes 011 3 bytes / Reserved Single-beat / Burst 100 4 bytes / Reserved 101 5 bytes / Reserved 110 6 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 00000External (system logic) 10000Processor (instruction) 10001Processor (data)
16–17	—	Reserved
18–19	TRANSTYPE	Transaction type for detected error 00 Snoop (tag/status read) 01 Write 10 Read 11 Read-modify-write
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).

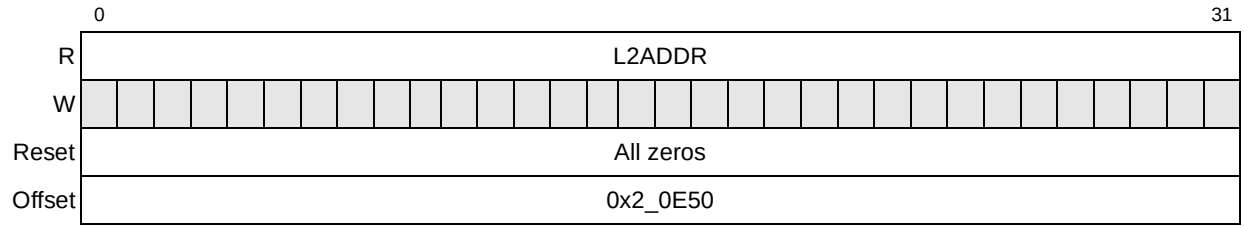


Figure 7-21. L2 Error Address Capture Register (L2ERRADDR)

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).

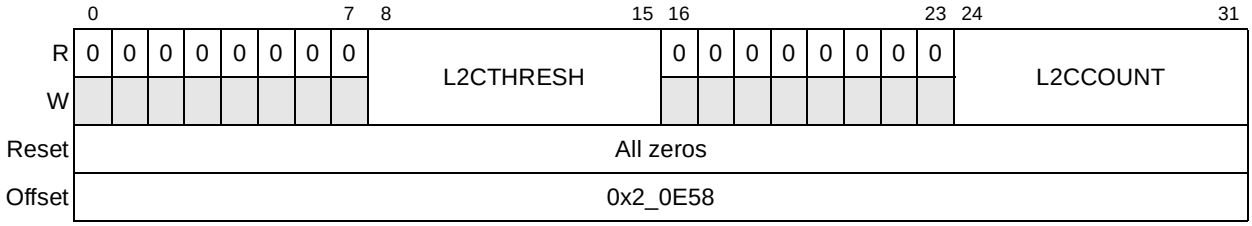


Figure 7-22. L2 Error Control Register (L2ERRCTL)

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
- 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 MPC8540 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 MPC8540 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.

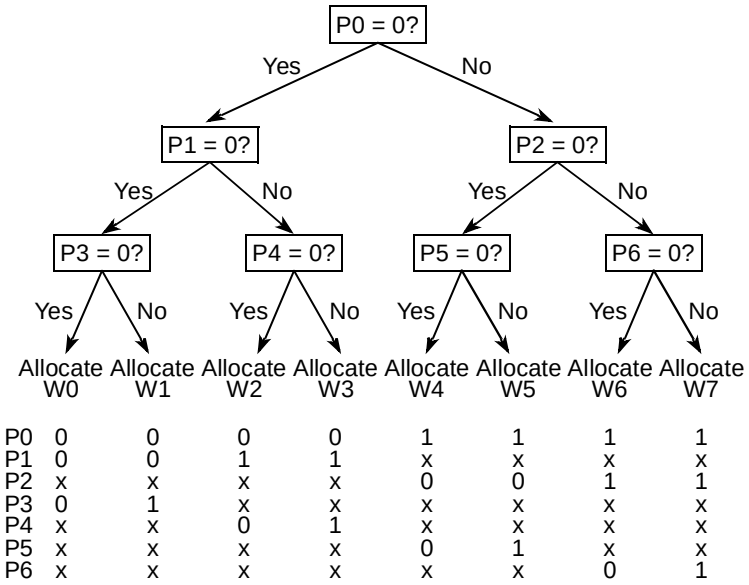


Figure 7-23. L2 Cache Line Replacement Algorithm

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						
	P 0	**P 1**	**P 2**	**P 3**	**P 4**	**P 5**	**P 6**
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 MPC8540 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 MPC8540.

  The ECM provides a mechanism for I/O-initiated transactions to snoop the core complex bus (CCB) of the e500 core in order to maintain coherency across cacheable local memory. It also provides a flexible, easily expandable switch-type structure for e500- and I/O-initiated transactions to be routed (dispatched) to target modules on the MPC8540.
- Chapter 9, "DDR Memory Controller," describes the DDR SDRAM memory controller of the MPC8540. This fully programmable controller supports most DDR memories available today, including both buffered and unbuffered devices. The built-in error checking and correction (ECC) ensures very low bit error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. A large set of special features like DLL software override, crawl mode, and ECC error injection support rapid system debug.
- Chapter 10, "Programmable Interrupt Controller," describes the embedded programmable interrupt controller (PIC) of the MPC8540. This controller is an OpenPIC-compliant interrupt controller that provides interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them, and delivering them to the CPU for servicing.
- Chapter 11, "I^2C Interface," describes the inter-IC (IIC or I^2C) bus controller of the MPC8540. This synchronous, serial, bidirectional, multi-master bus allows two-wire connection of devices such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs. The MPC8540 powers up in boot sequencer mode, which allows the I^2C controller to initialize configuration registers.
- Chapter 12, "DUART," describes the (dual) universal asynchronous receiver/transmitters (UARTs) which feature a PC16552D-compatible programming model. These independent UARTs are provided specifically to support system debugging.
- Chapter 13, "Local Bus Controller," describes the local bus controller of the MPC8540. The main component of the local bus controller (LBC) is its memory controller which provides a seamless interface to many types of memory devices and peripherals. The

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

- Chapter 14, "Three-Speed Ethernet Controllers," describes the two three-speed Ethernet controllers on the MPC8540. 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 MPC8540 memory coherency module.
- Chapter 15, "DMA Controller," describes the four-channel general-purpose DMA controller of the MPC8540. The DMA controller transfers blocks of data, independent of the e500 core or external hosts. Data movement occurs among RapidIO and the local address space. The DMA controller has four high-speed channels. Both the e500 core and external masters can initiate a DMA transfer. All channels are capable of complex data movement and advanced transaction chaining.
- Chapter 16, "PCI/PCI-X Bus Interface," describes the PCI/PCI-X controller of the MPC8540.
- Chapter 17, "RapidIO Interface," describes the RapidIO controller of the MPC8540.

Chapter 8
e500 Coherency Module

8.1 Introduction

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.

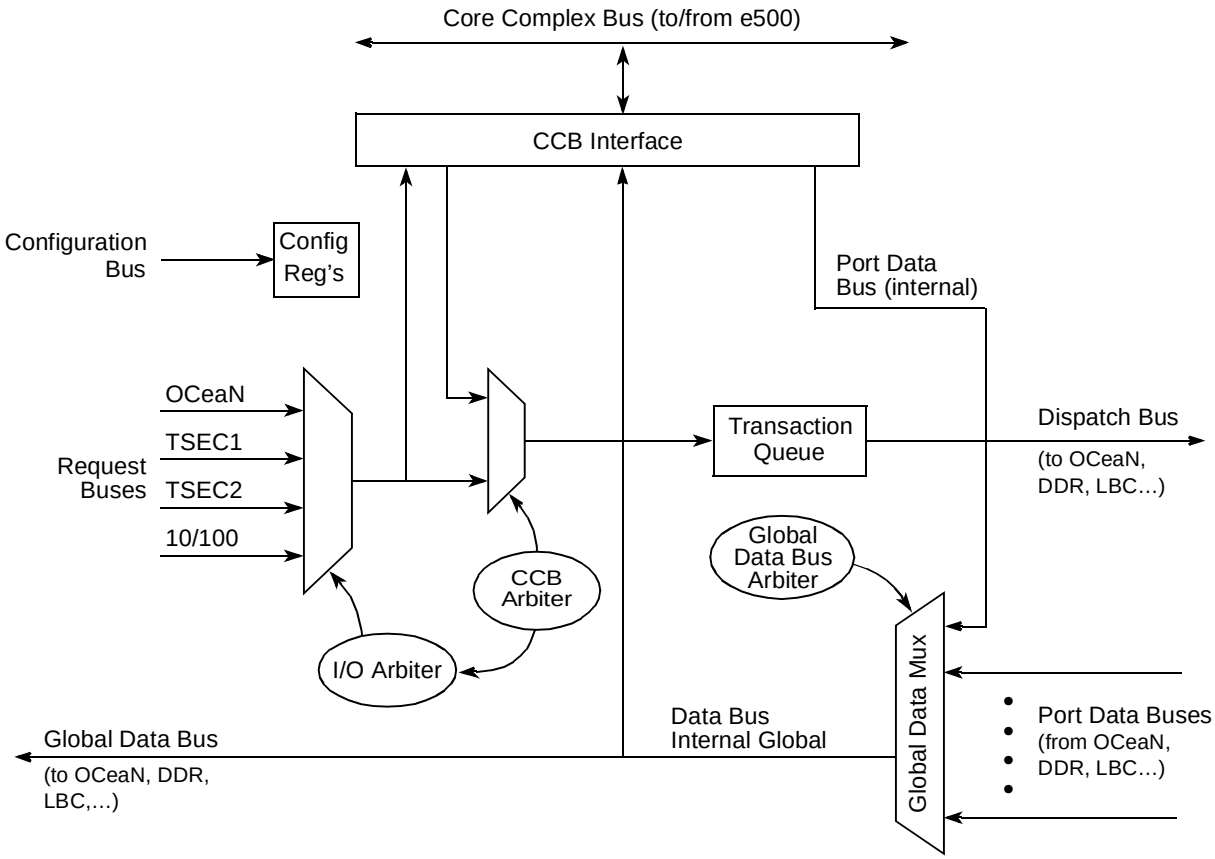


Figure 8-1. e500 Coherency Module Block Diagram

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 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.

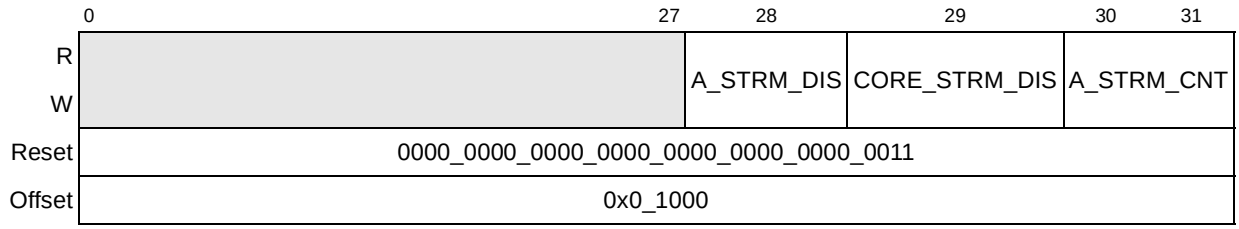


Figure 8-2. ECM CCB Address Configuration Register (EEBACR)

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.

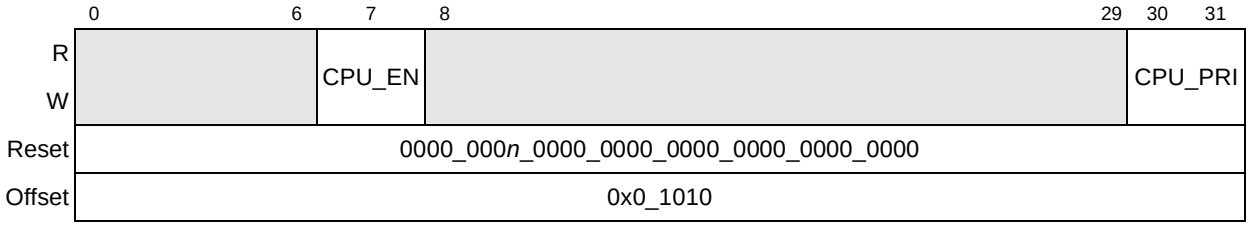


Figure 8-3. ECM CCB Port Configuration Register (EEBPCR)

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.

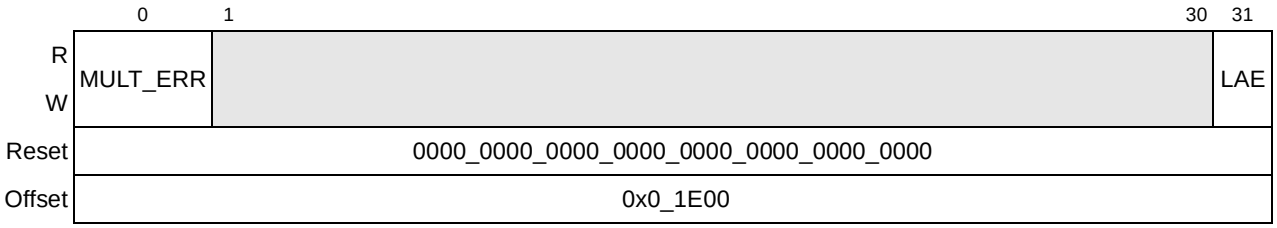


Figure 8-4. ECM Error Detect Register (EEDR)

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.

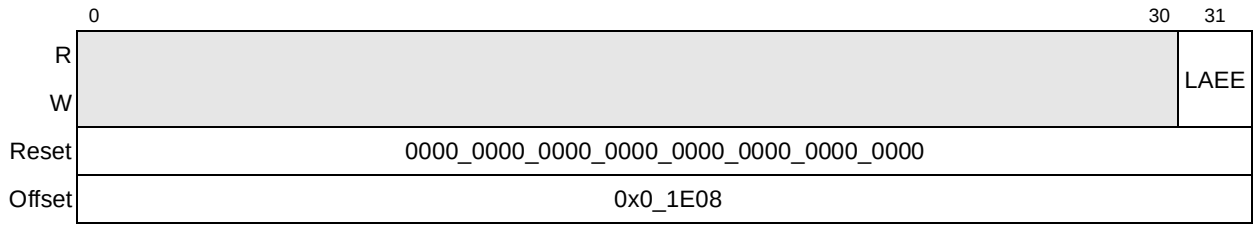


Figure 8-5. ECM Error Enable Register (EEER)

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.

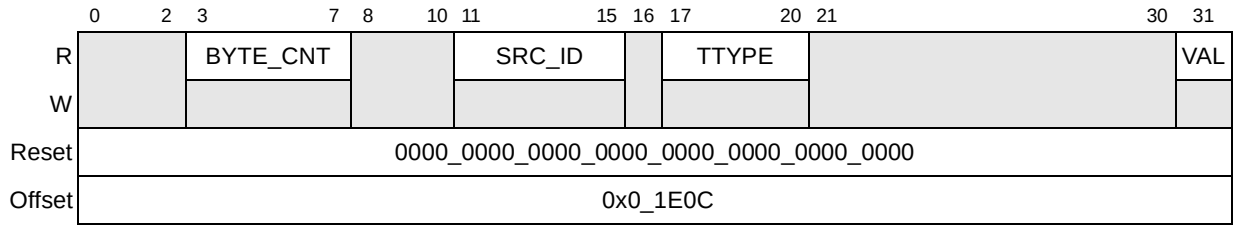


Figure 8-6. ECM Error Attributes Capture Register (EEATR)

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 Reserved 10101 DMA 10110 Reserved 10111 SAP 11000 TSEC1 11001 TSEC2 11010 10/100 FEC 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.

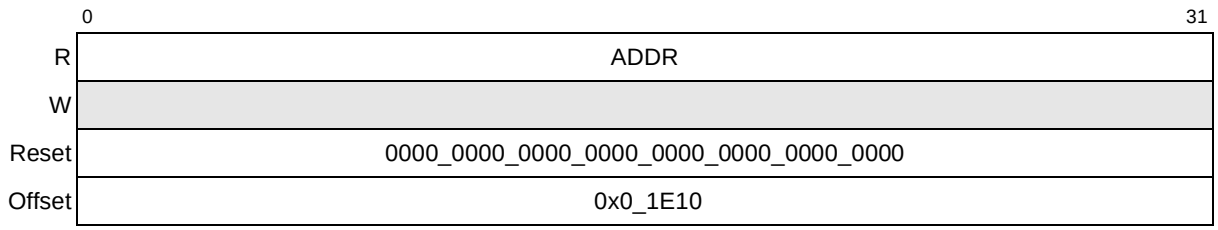


Figure 8-7. ECM Error Address Capture Register (EEADR)

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. 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 MPC8540, the CPU boot configuration power-on reset pin should be pulled high to initially set EEBPCR[CPU_EN]. See Chapter 4, “Reset, Clocking, and Initialization,” for more information on power-up reset initialization.

If any device other than the e500 core, such as the boot sequencer or PCI, is used to initialize the MPC8540, the CPU boot configuration power-on reset pin should be pulled low to initially clear EEBPCR[CPU_EN]. This prevents the e500 core from accessing any configuration registers or local memory space during initialization. However, in any such system, one step near the end of the initialization routine must set EEBPCR[CPU_EN] to re-enable the e500 core. 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.

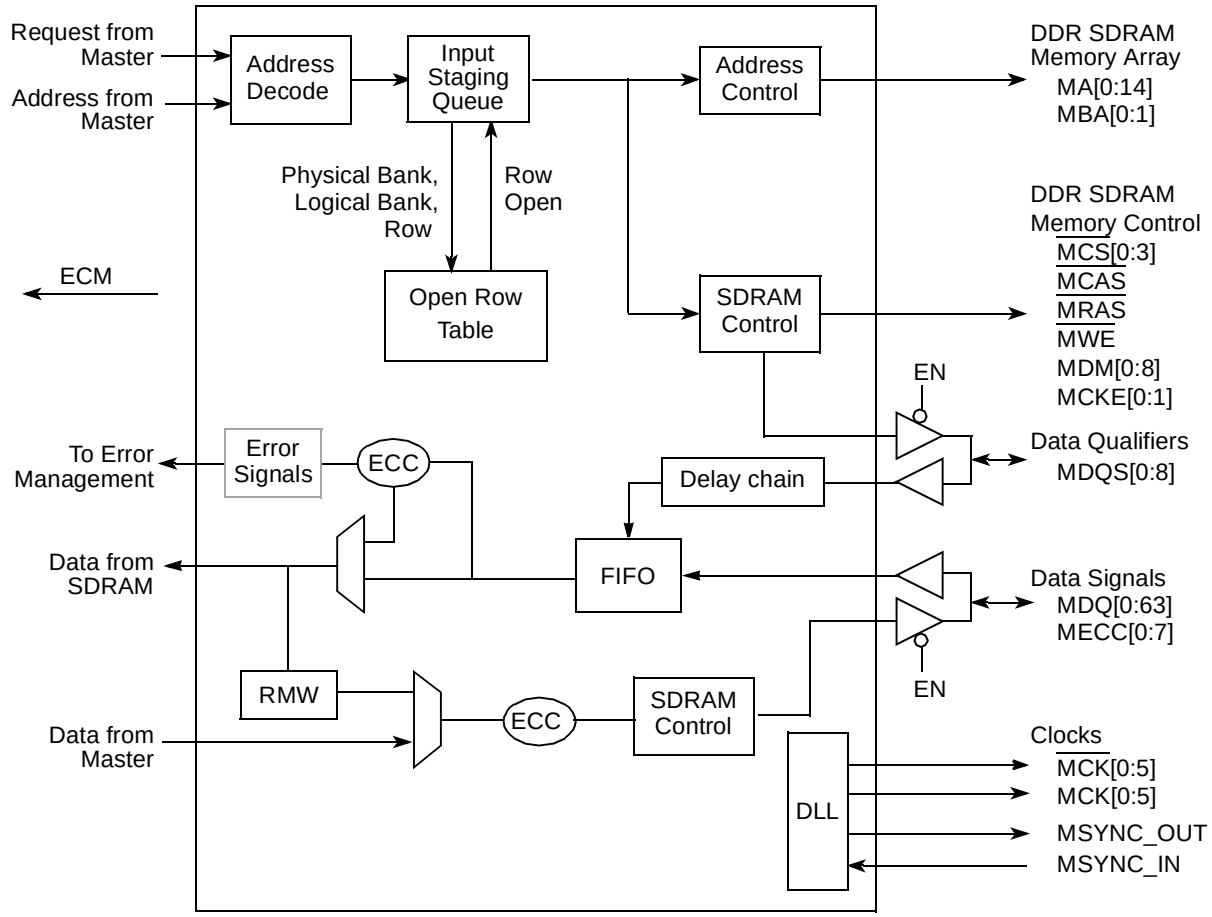


Figure 9-1. DDR Memory Controller Simplified Block Diagram

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 *MPC8540 Integrated Host 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 Description (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 20.4.3.2, "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	
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 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—MRAS is always driven unless the memory controller is idle.
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 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
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 MWE for various other SDRAM commands. Negated—Indicates a memory read operation
		Timing	Assertion/Negation—Similar timing as MRAS and MCAS. Used for write commands. High impedance—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], 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 MPC8540. 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 MCK. MCK/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 MPC8540. 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 20.4.3, "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.

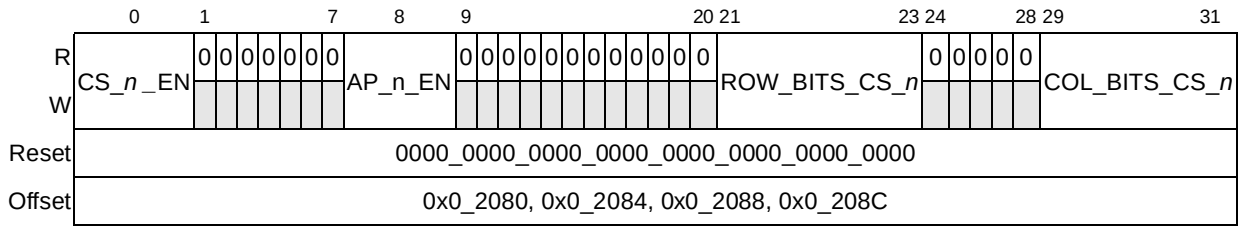


Figure 9-3. Chip Select Configuration Register (CS*n*_CONFIG)

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.

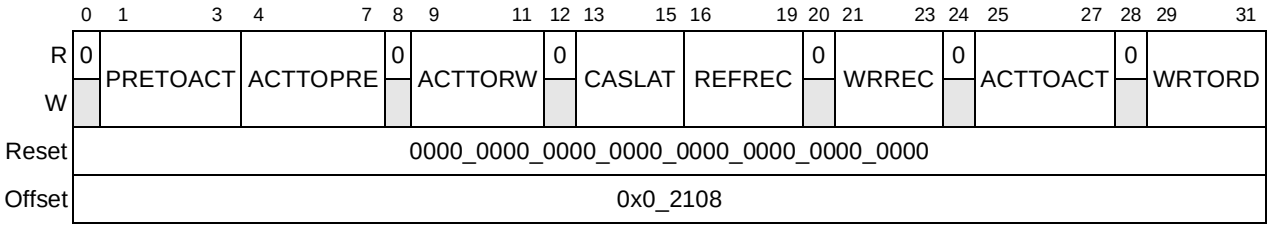


Figure 9-4. DDR SDRAM Timing Configuration Register 1 (TIMING_CFG_1)

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.

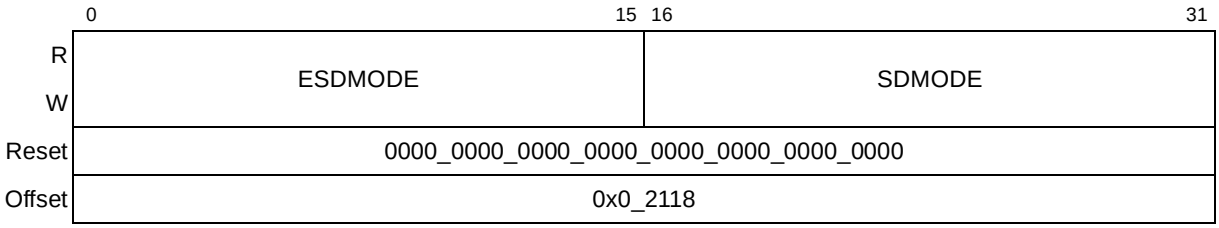


Figure 9-7. DDR SDRAM Mode Configuration Register (DDR_SDRAM_MODE)

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.

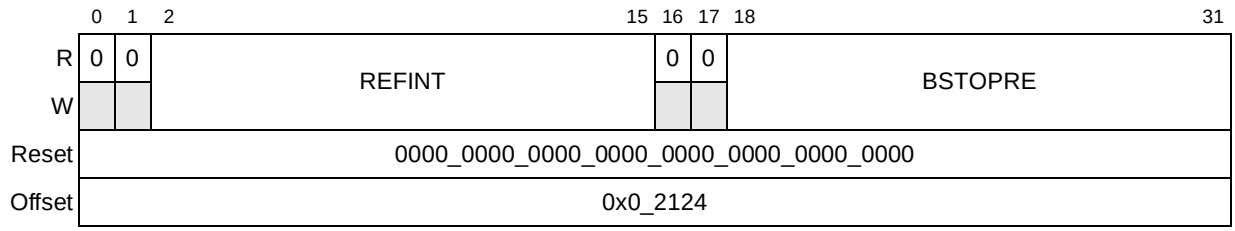


Figure 9-8. DDR SDRAM Interval Configuration Register (DDR_SDRAM_INTERVAL)

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 18.5.1.5.3, “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.

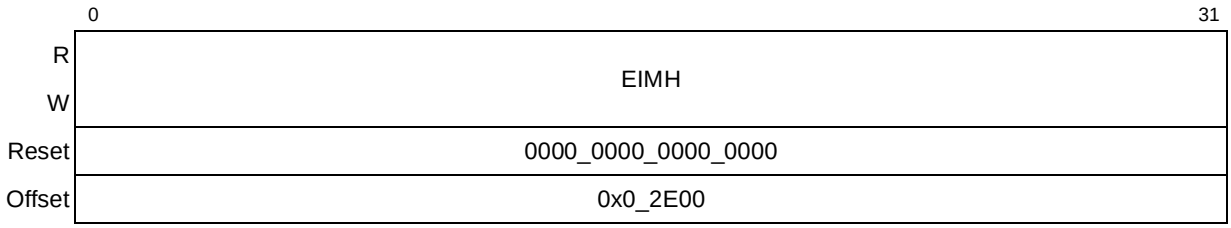


Figure 9-9. Memory Data Path Error Injection Mask High Register (DATA_ERR_INJECT_HI)

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.

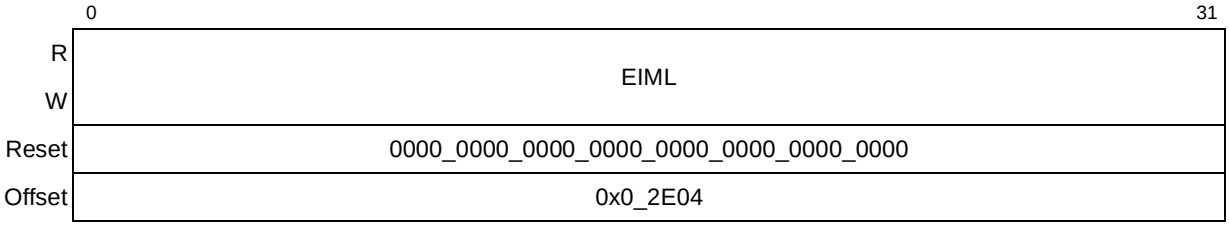


Figure 9-10. Memory Data Path Error Injection Mask Low Register (DATA_ERR_INJECT_LO)

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.

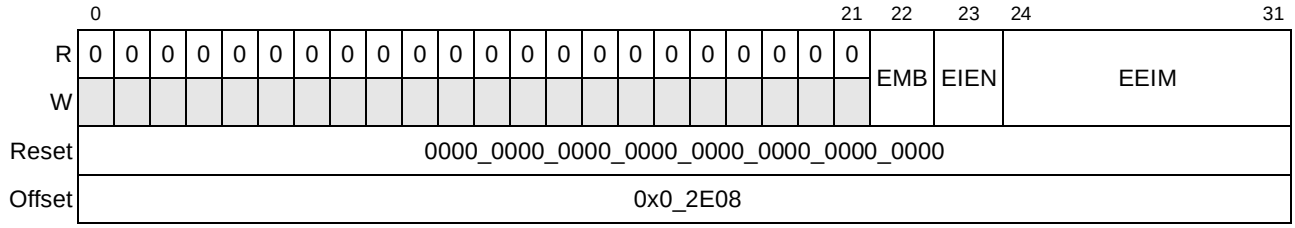


Figure 9-11. Memory Data Path Error Injection Mask ECC Register (ECC_ERR_INJECT)

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.

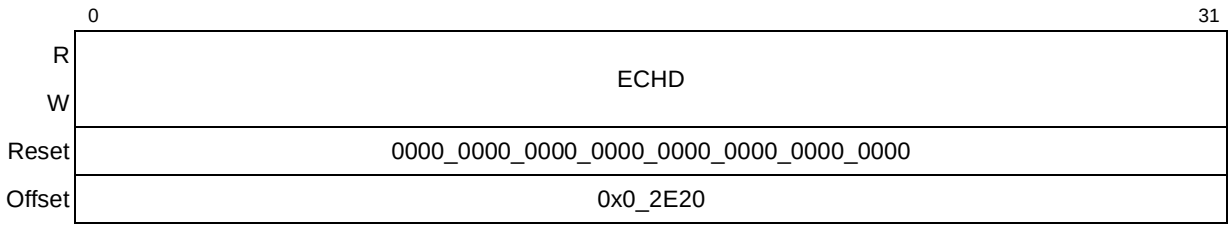


Figure 9-12. Memory Data Path Read Capture High Register (CAPTURE_DATA_HI)

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.

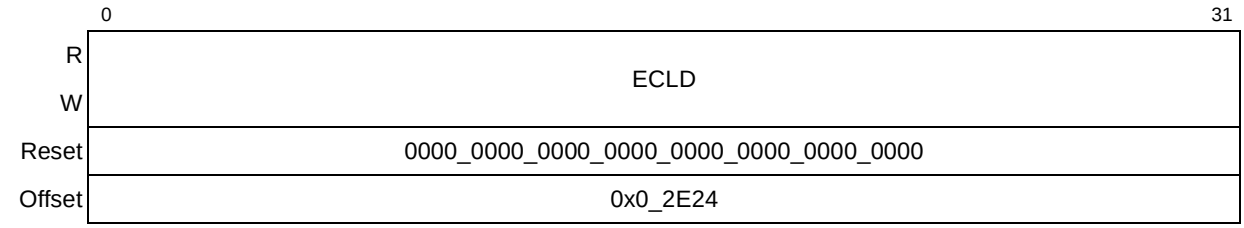


Figure 9-13. Memory Data Path Read Capture Low Register (CAPTURE_DATA_LO)

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.

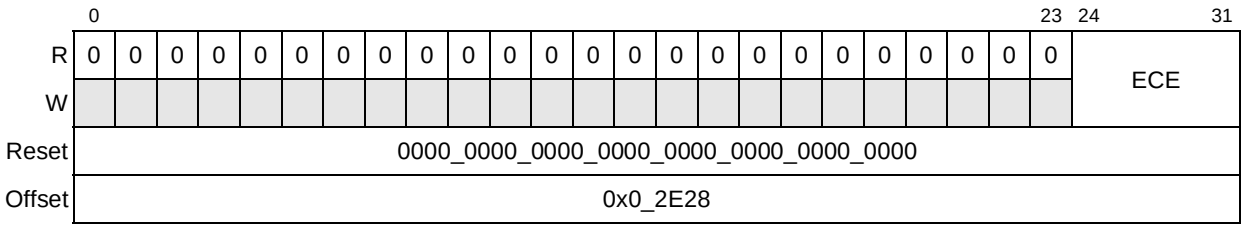


Figure 9-14. Memory Data Path Read Capture ECC Register (CAPTURE_ECC)

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.

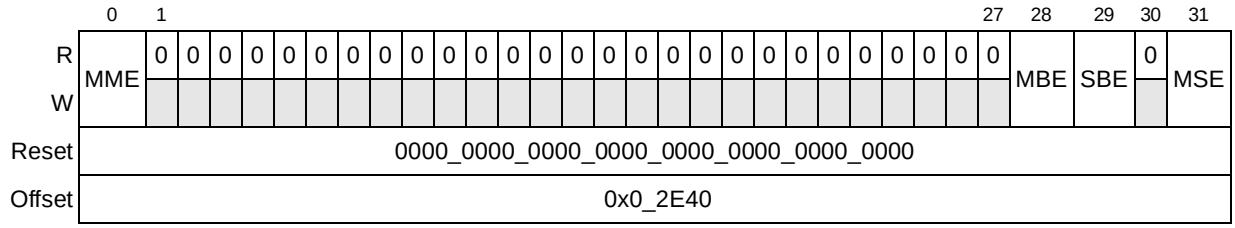


Figure 9-15. Memory Error Detect Register (ERR_DETECT)

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.

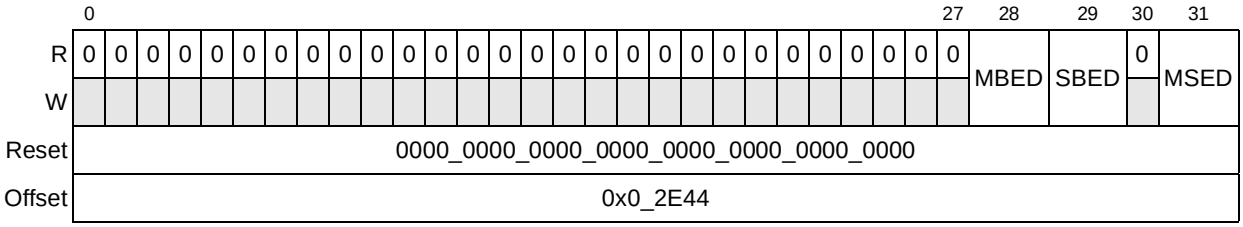


Figure 9-16. Memory Error Disable Register (ERR_DISABLE)

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).

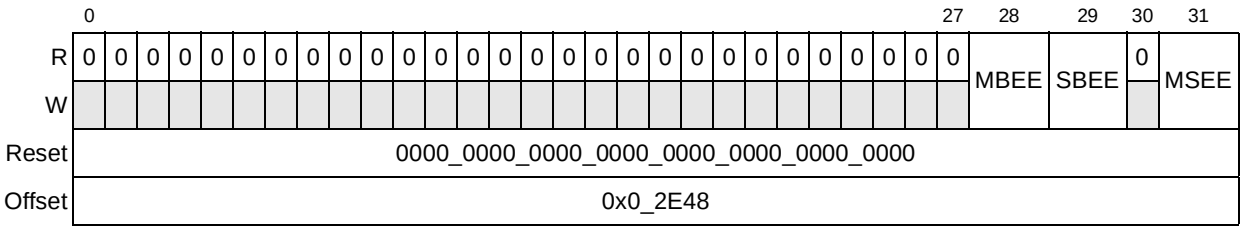


Figure 9-17. Memory Error Interrupt Enable Register (ERR_INT_EN)

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.

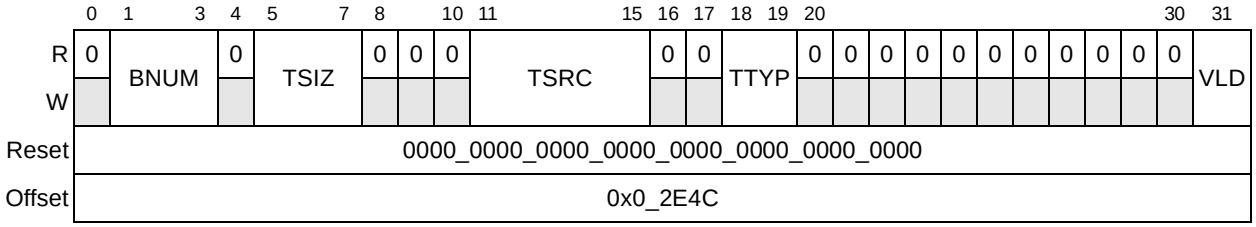


Figure 9-18. Memory Error Attributes Capture Register (CAPTURE_ATTRIBUTES)

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 Reserved 10101 DMA 10110 Reserved 10111 SAP 11000 TSEC1 11001 TSEC2 11010 FEC 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.

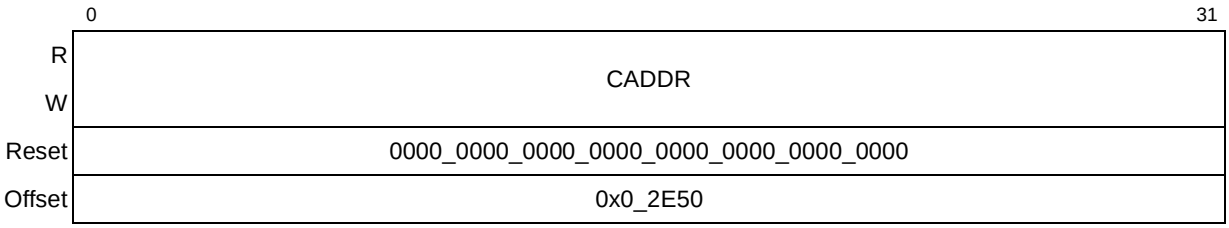


Figure 9-19. Memory Error Address Capture Register (CAPTURE_ADDRESS)

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.

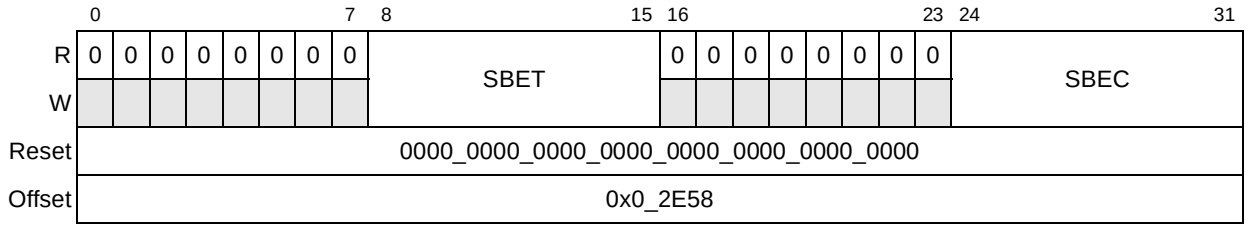


Figure 9-20. Single-Bit ECC Memory Error Management Register (ERR_SBE)

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.

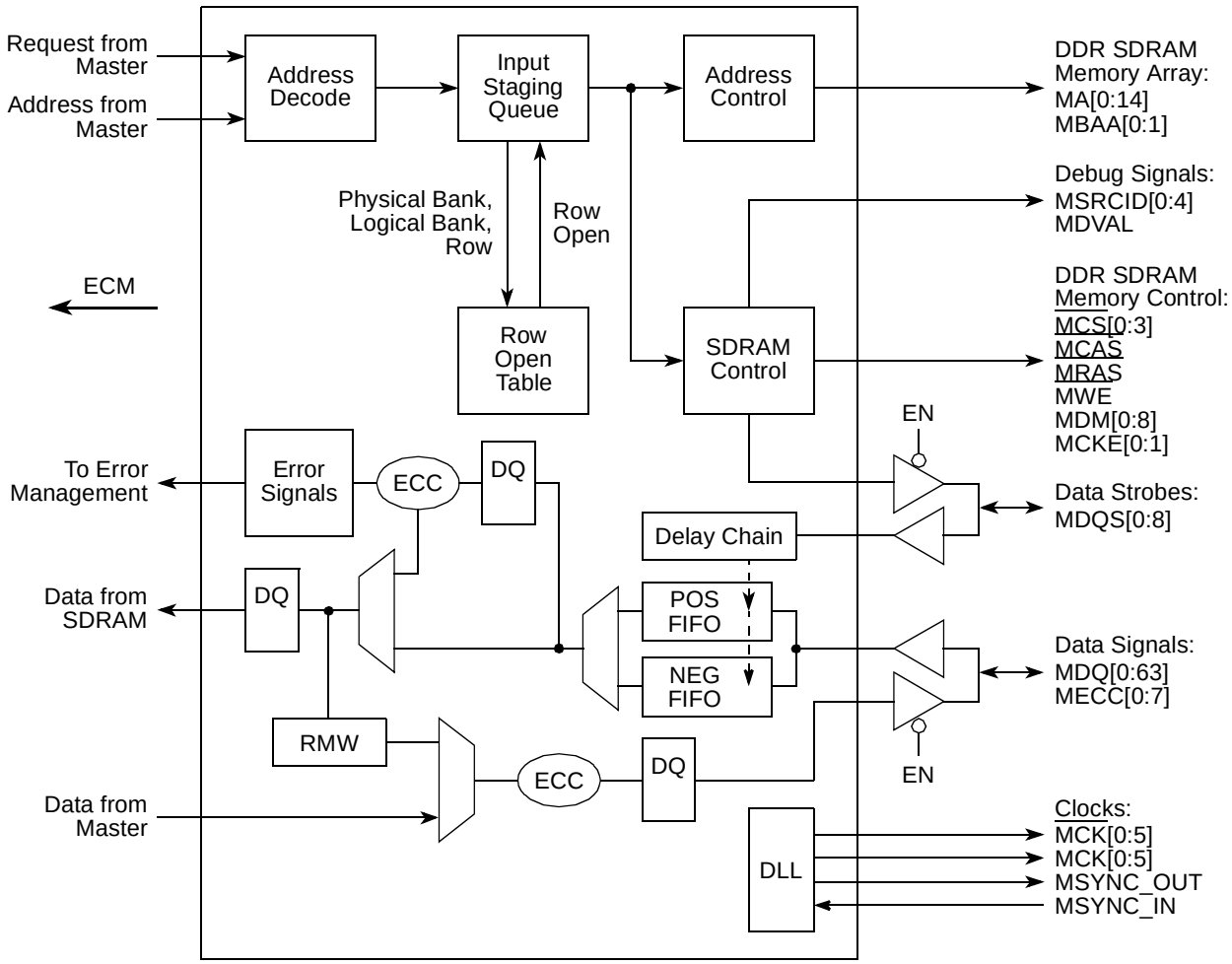


Figure 9-21. DDR Memory Controller Block Diagram

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.

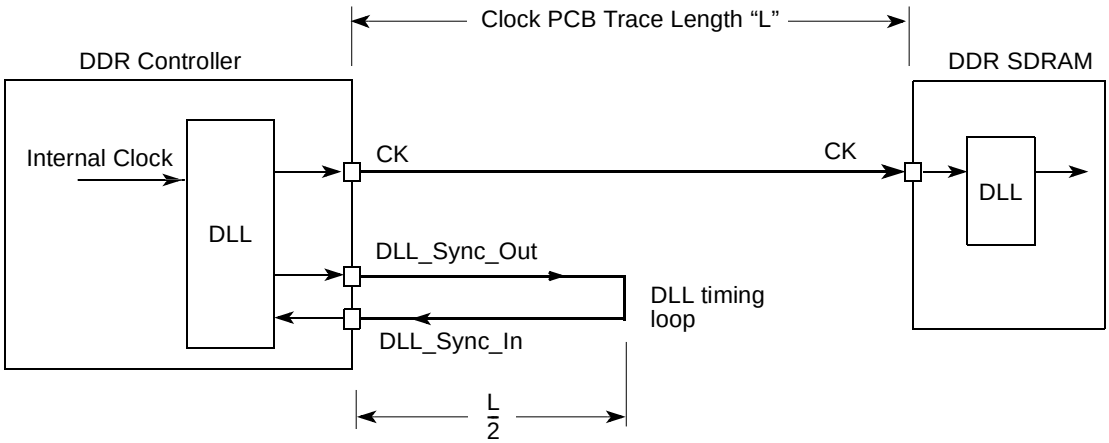


Figure 9-22. Controller DLL Timing Loop

Figure 9-23 shows an example DDR SDRAM configuration with four physical banks.

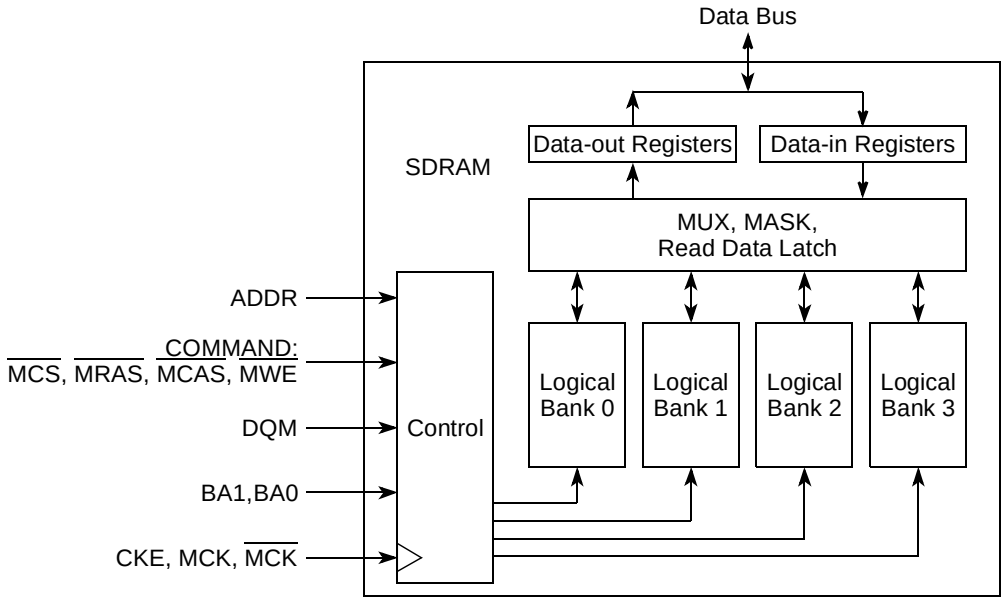


Figure 9-23. Typical Dual Data Rate SDRAM Internal Organization

Figure 9-24 shows some typical signal connections.

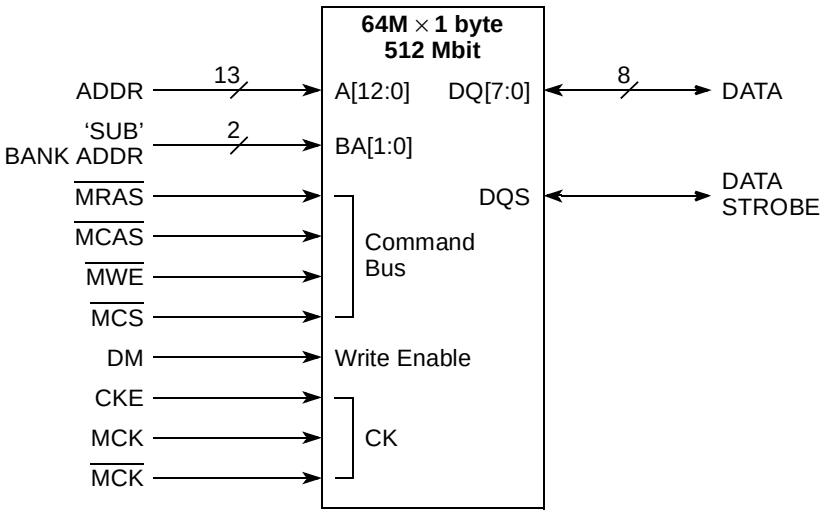
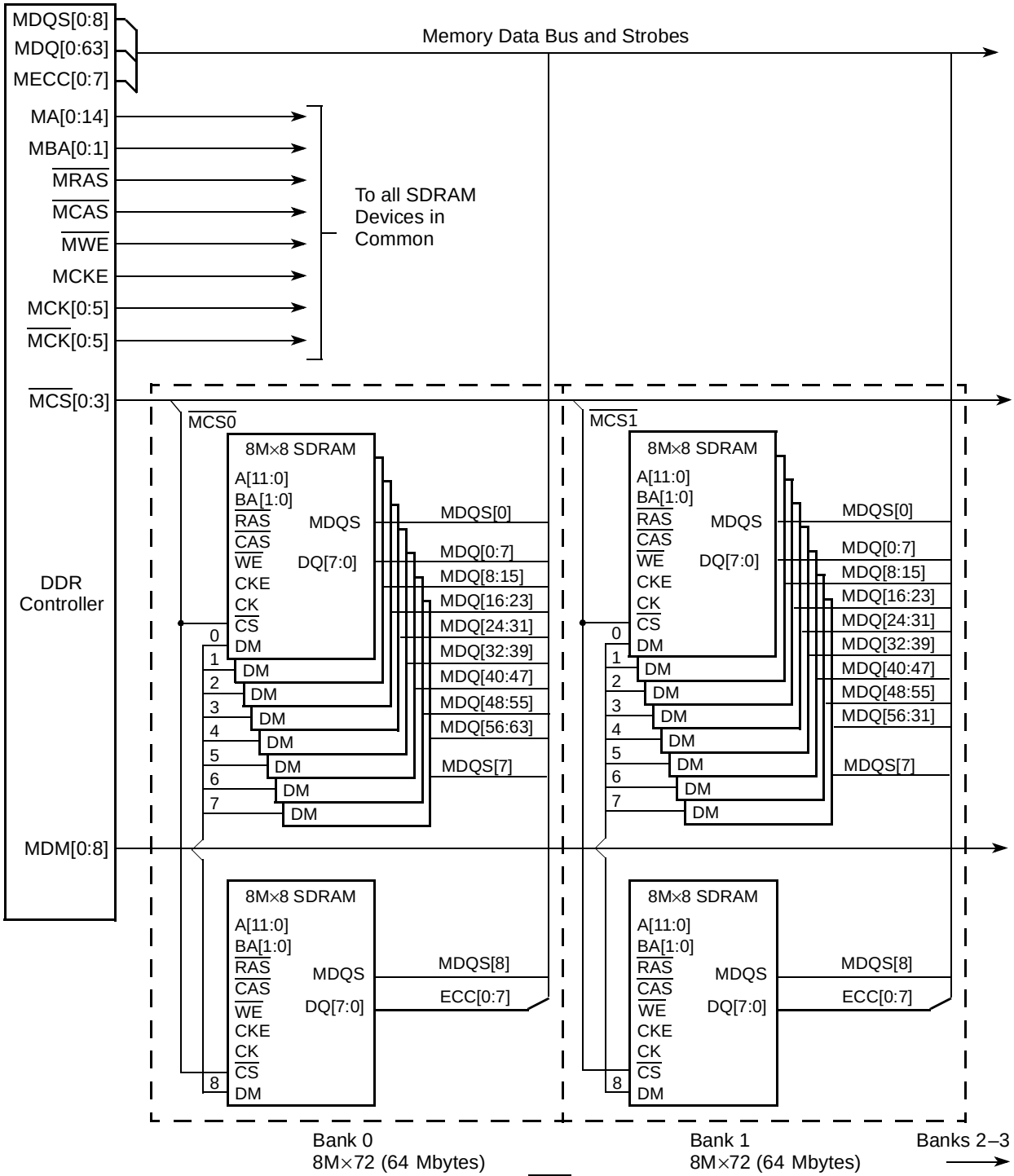


Figure 9-24. Typical DDR SDRAM Interface Signals

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 MPC8540 device's AC timing specifications, desired memory operating frequency, capacitive loads, and board routing loads, can assist the system designer in deciding signal buffering requirements. The MPC8540 DDR memory controller drives 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 MPC8540 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.

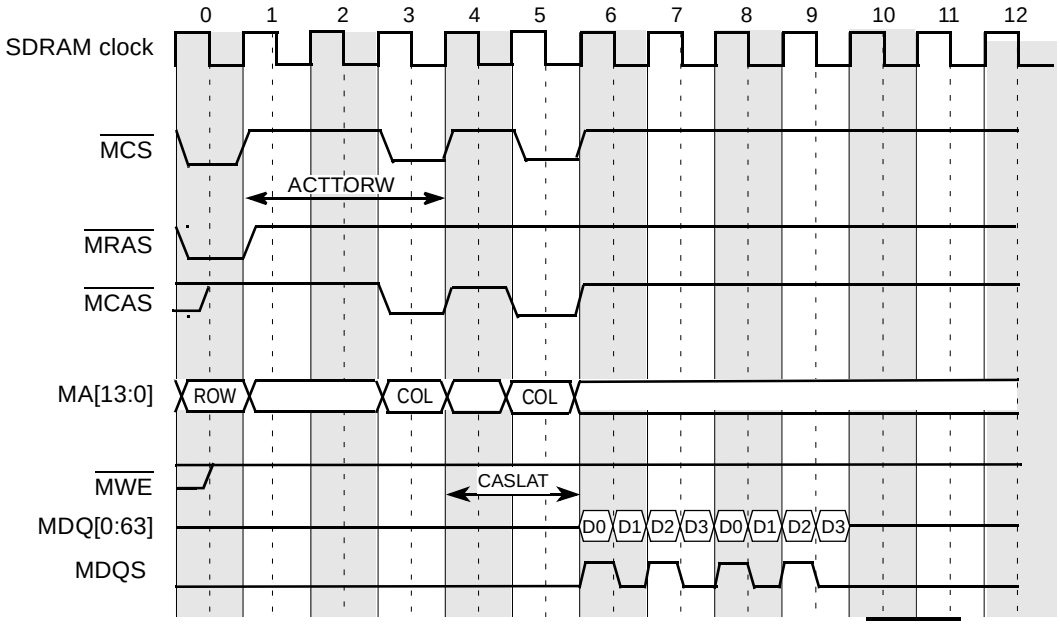


Figure 9-26. DDR SDRAM Burst Read Timing—ACTTORW = 3, MCAS Latency = 2

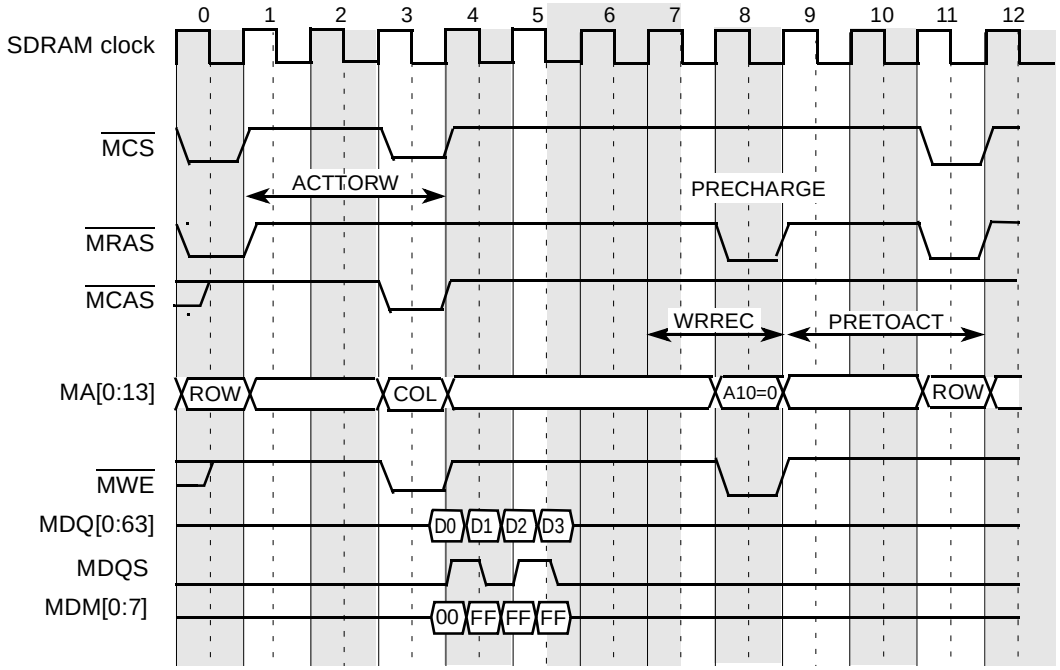


Figure 9-27. DDR SDRAM Single-Beat (Double-Word) Write Timing—ACTTORW = 3

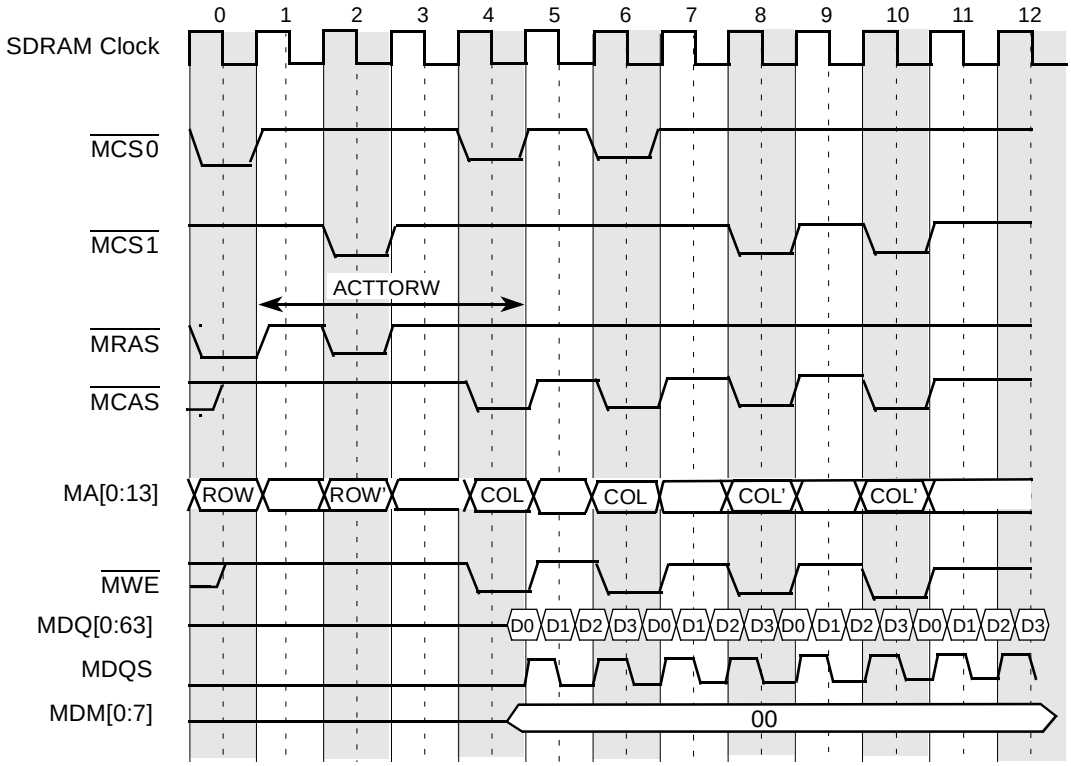


Figure 9-28. DDR SDRAM Burst Write Timing—ACTTORW = 4

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.

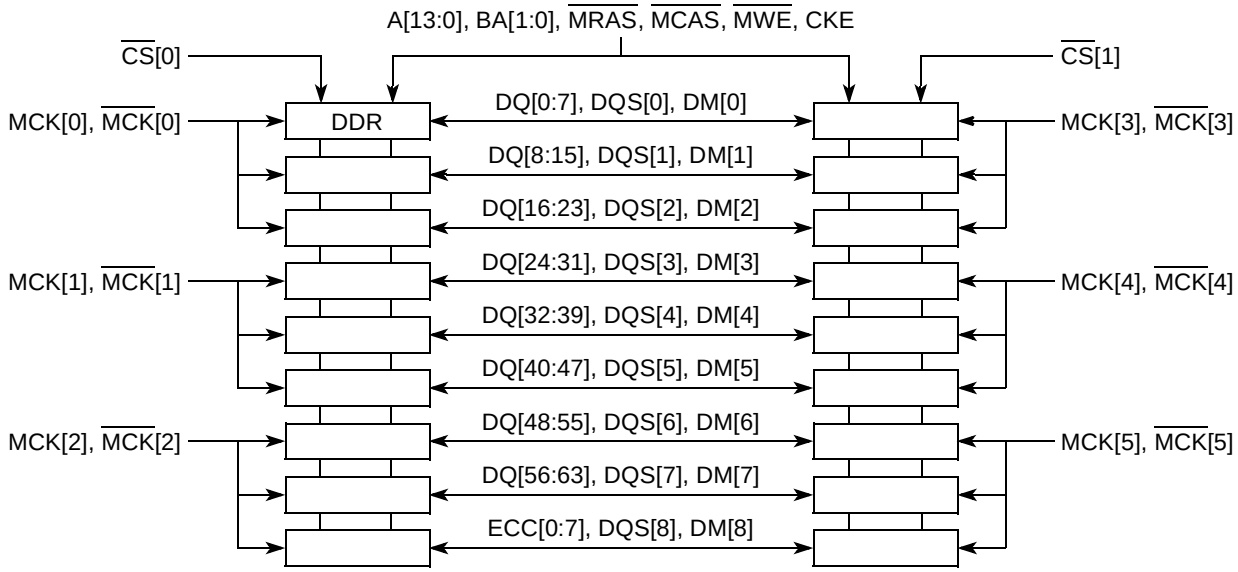


Figure 9-29. DDR SDRAM Clock Distribution Example

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.

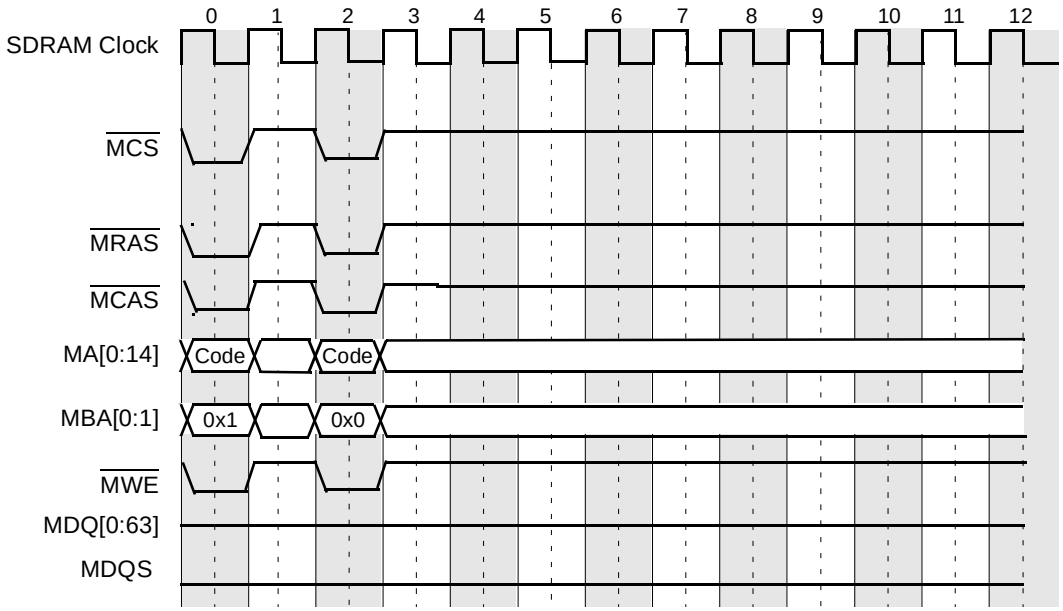


Figure 9-30. DDR SDRAM Mode-Set Command Timing

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.

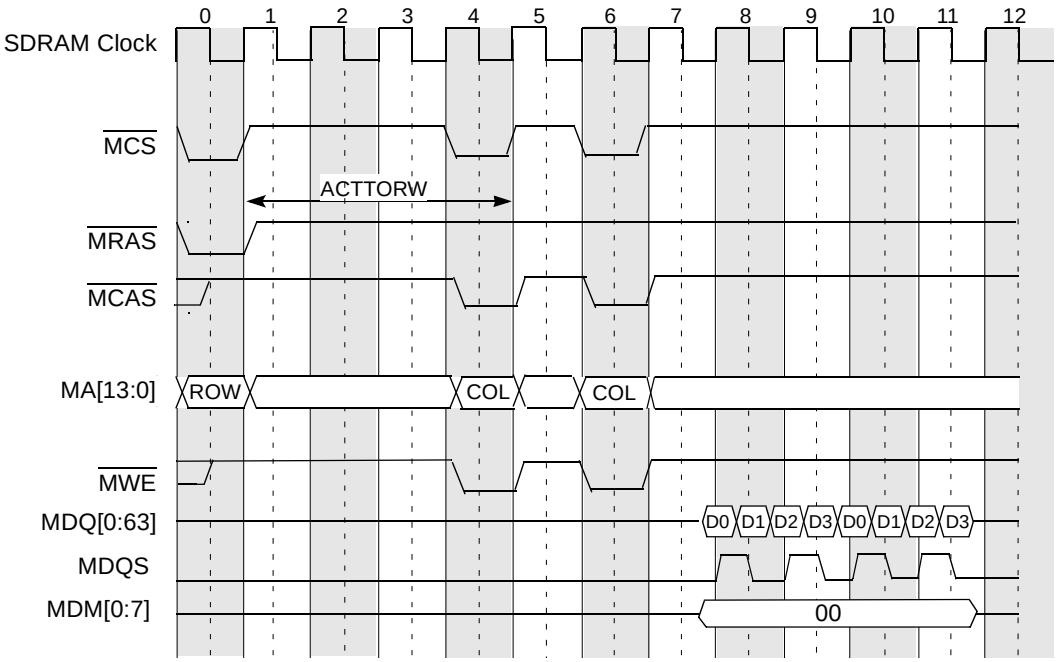


Figure 9-31. Registered DDR SDRAM DIMM Burst Write Timing

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

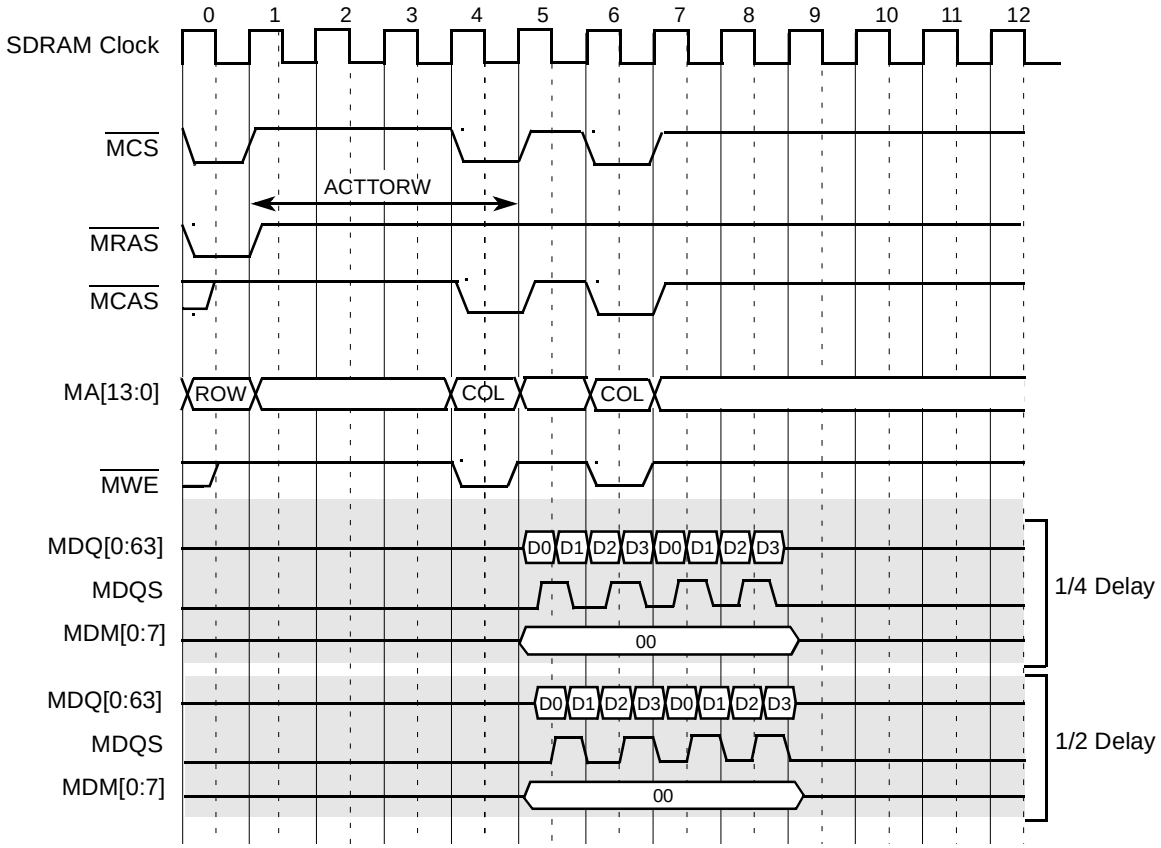


Figure 9-32. Write Timing Adjustments Example

9.5.8 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).

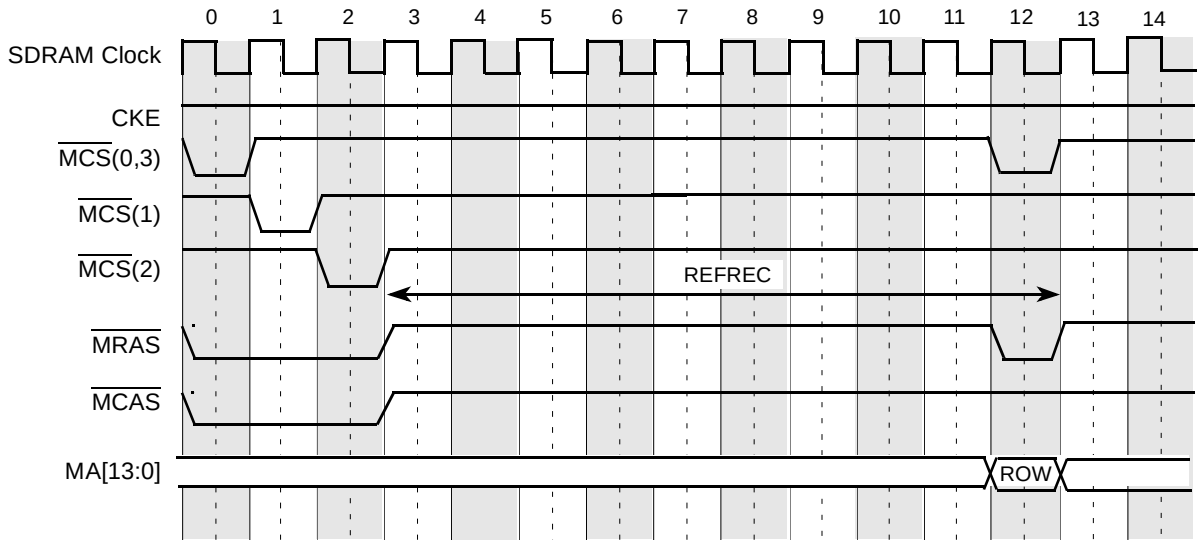


Figure 9-33. DDR SDRAM Bank-Staggered Auto-Refresh Timing

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.

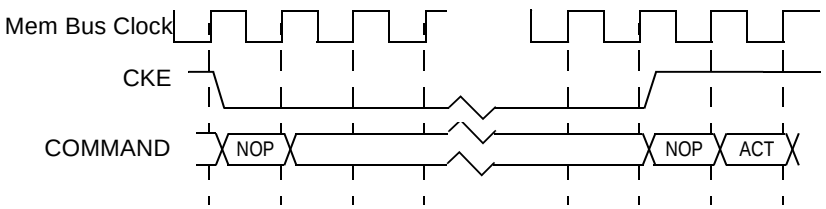


Figure 9-34. DDR SDRAM Power-Down Mode

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.

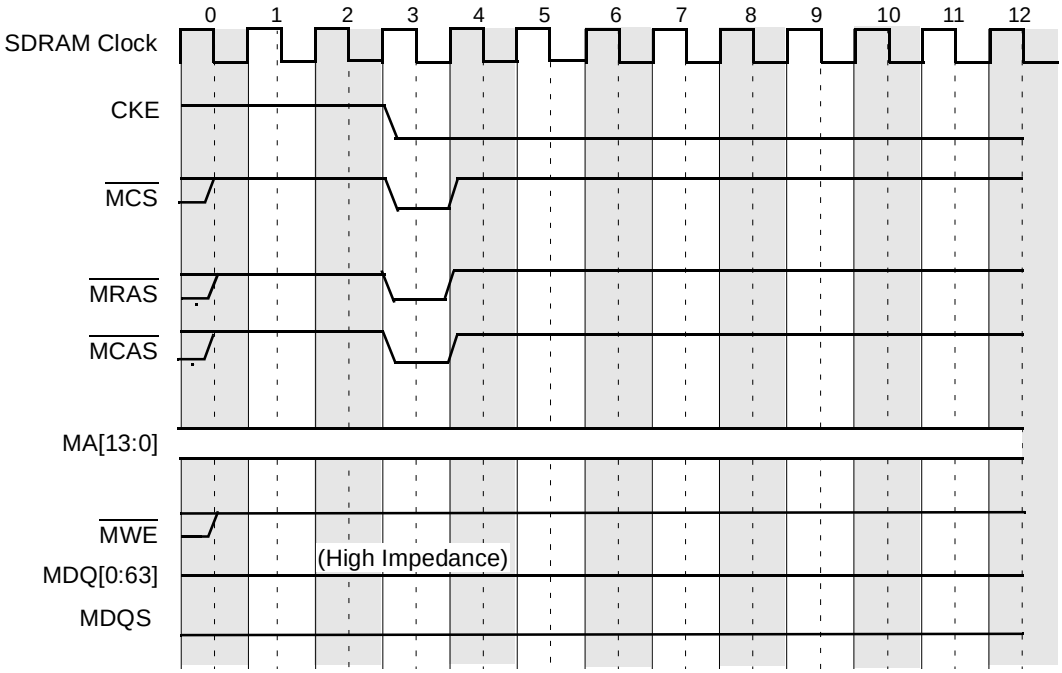


Figure 9-35. DDR SDRAM Self-Refresh Entry Timing

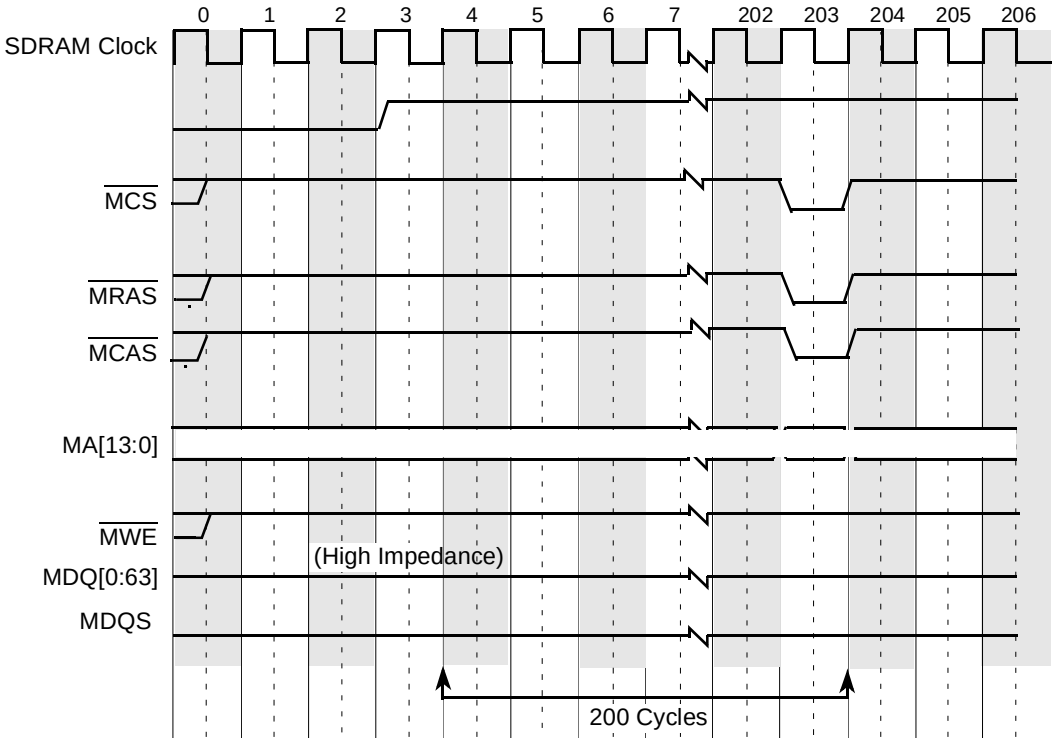


Figure 9-36. DDR SDRAM Self-Refresh Exit Timing

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							
	0	1	2	3	4	5	6	7
0	•	•						•
1	•		•					•
2	•			•				•

Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
32			•	•				•
33			•		•			•
34	•		•		•			

Table 9-32. DDR SDRAM ECC Syndrome Encoding (continued)

Data Bit	Syndrome Bit								Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
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 MPC8540 showing the relationship of the various functional blocks and external signals to the PIC unit is shown in Figure 10-1.

The PIC unit receives interrupt signals from 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 10/100 Ethernet controller, the DUART, 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.

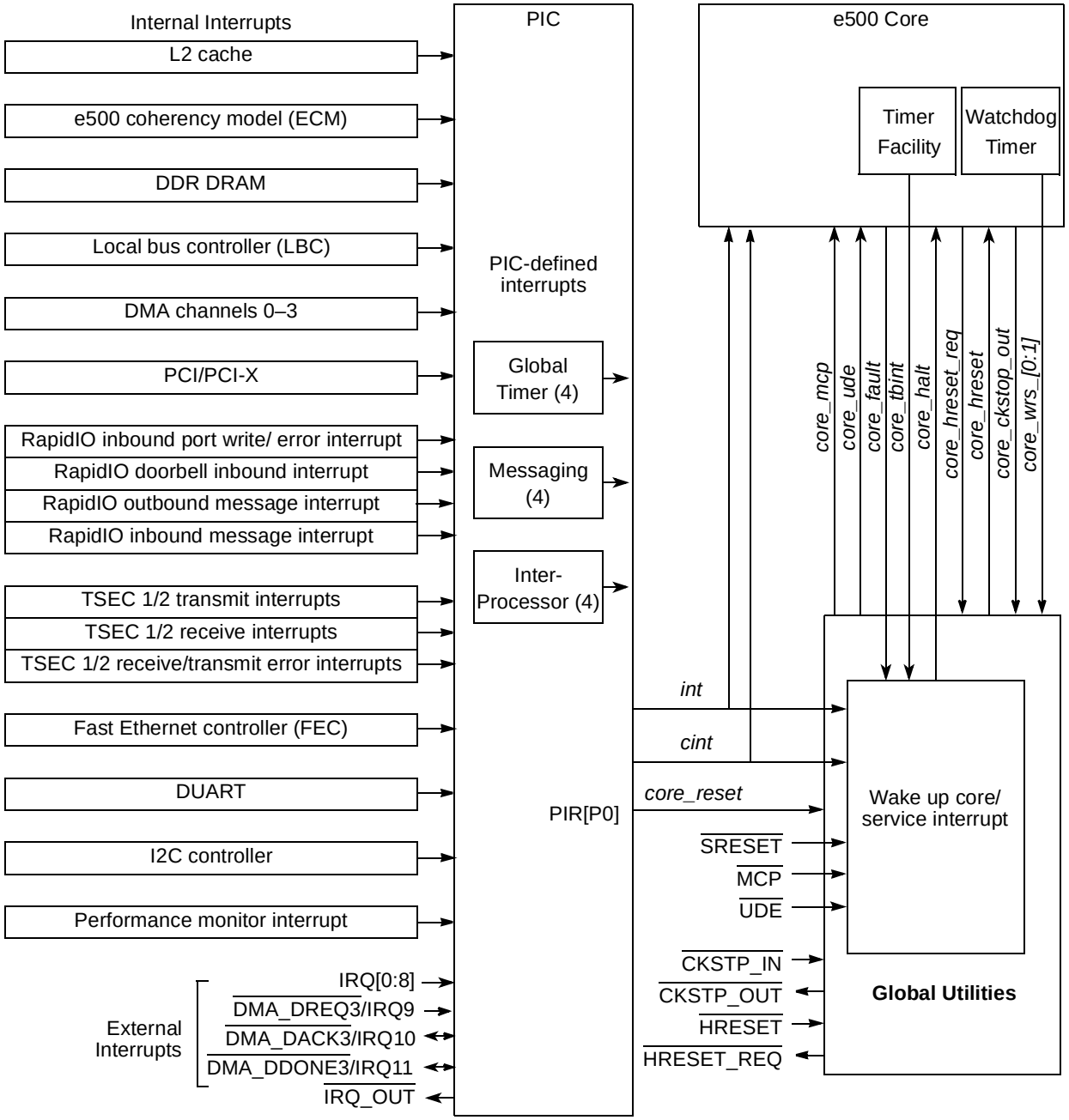


Figure 10-1. MPC8540 Interrupt Sources Block Diagram

10.1.2 Features

- Programming model compliant with the OpenPIC architecture
- Support for 12 external and 23 internal interrupt sources. Serial interrupts are not supported.
- Four interprocessor interrupt channels

- Four 32-bit messaging interrupt channels
- Four global high resolution timers 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 MPC8540 conditions (shown in Table 10-2) cause interrupts to the core (and wake up the core when it is in a low-power state).

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* (MPC8540 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 Decrementer	*core_tbint*	The source of both of these interrupts is the time base facility within the e500 core. The MPC8540 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 18, "Global Utilities."
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 MPC8540 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.

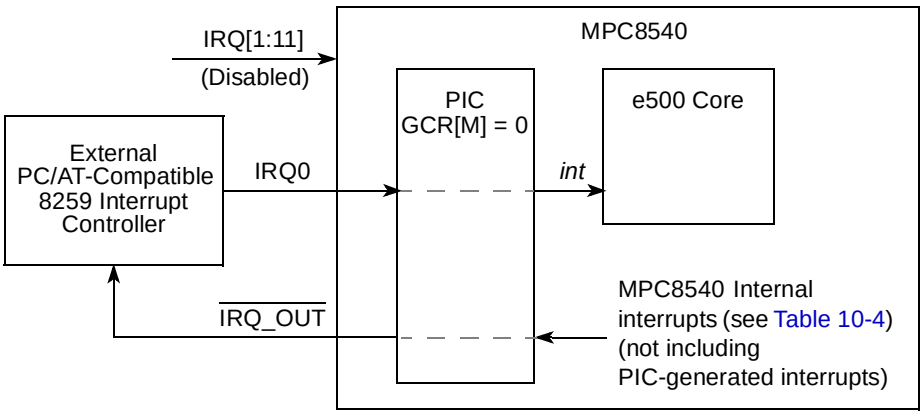


Figure 10-2. Pass-Through Mode Example

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 47 separate interrupts from five different sources as follows:

- 12 external—Off-chip signals, IRQ[0:11]
- 23 internal—On-chip. Sources are L2, ECM, DDR, local bus controller, DMA, PCI/PCI-X, RapidIO, TSEC, FEC, DUART, 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 MPC8540
- 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 23 internal interrupt sources for the MPC8540. 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	Fast Ethernet controller (FEC)
8	PCI/PCI-X	26	DUART
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	Unused
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 18.4.1.10, “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 18.4.1.13, "Machine Check Summary Register (MCPSUMR)," Table 10-2 and Table 10-3.	
		State Meaning	Asserted—The MPC8540 should initiate a machine check interrupt or enter the checkstop state as directed by the MSR. Negated—Machine check handling is not being requested by the external system.
		Timing	Assertion—May occur at any time, asynchronous to any clock. Negation—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 MPC8540 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	10.3.2.6/10-24
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	10.3.4.2/10-30
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	10.3.4.2/10-30
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	10.3.4.2/10-30
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	10.3.4.2/10-30
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	10.3.5.2/10-31
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.

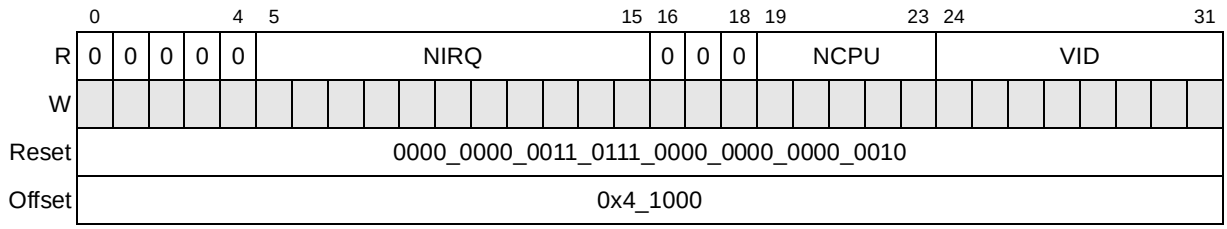


Figure 10-3. Feature Reporting Register (FRR)

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 23 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 MPC8540 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.

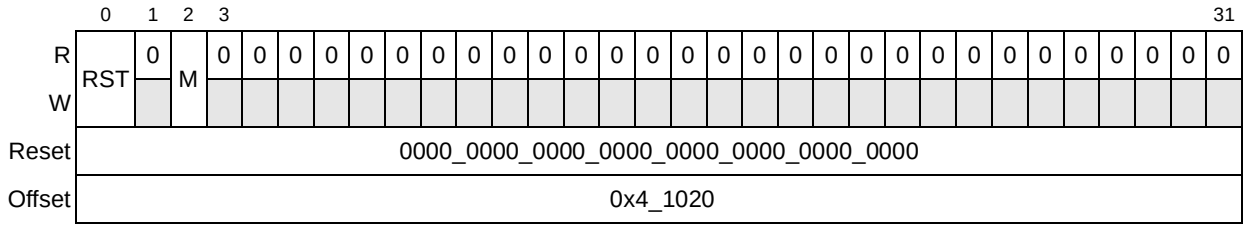


Figure 10-4. Global Configuration Register (GCR)

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.

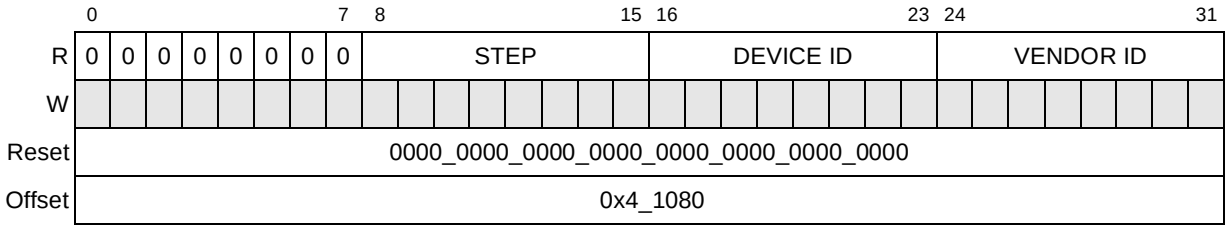


Figure 10-5. Vendor Identification Register (VIR)

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.

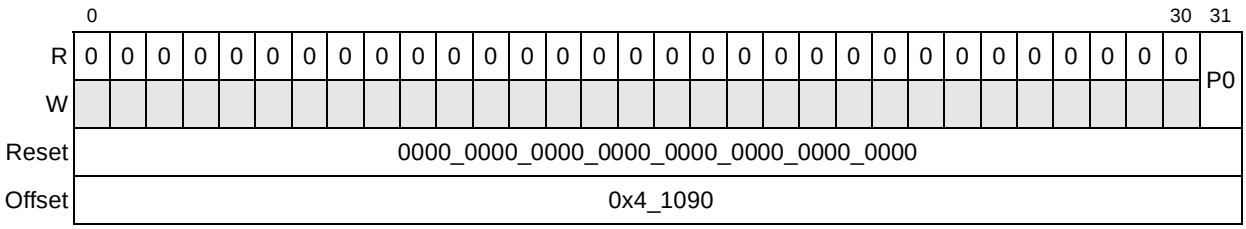


Figure 10-6. Processor Initialization Register (PIR)

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	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.

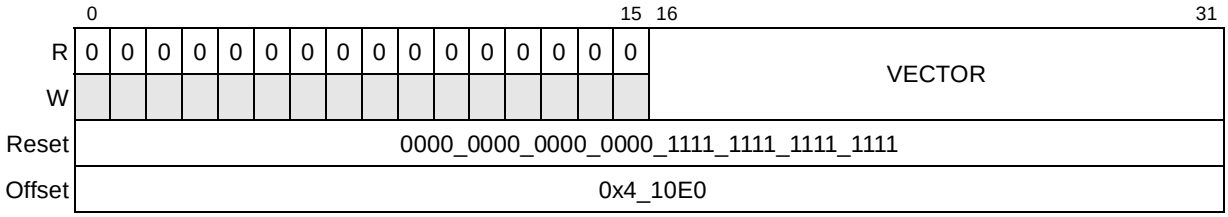


Figure 10-8. Spurious Vector Register (SVR)

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.

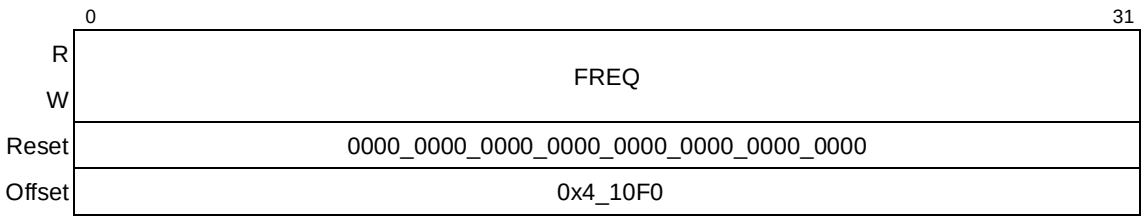


Figure 10-9. Timer Frequency Reporting Register (TFRR)

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.

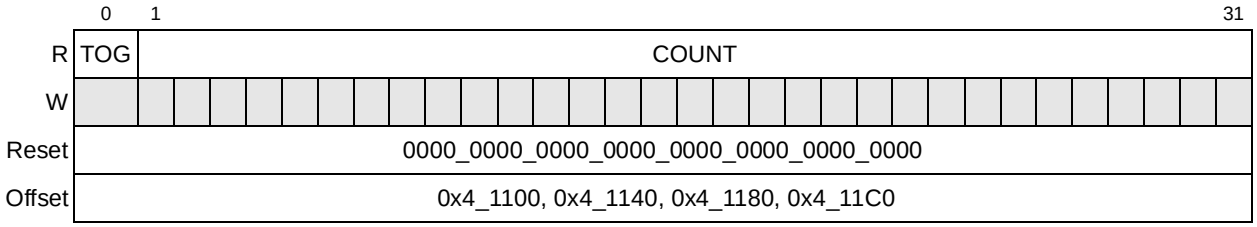


Figure 10-10. Global Timer Current Count Registers (GTCCR*n*)

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.

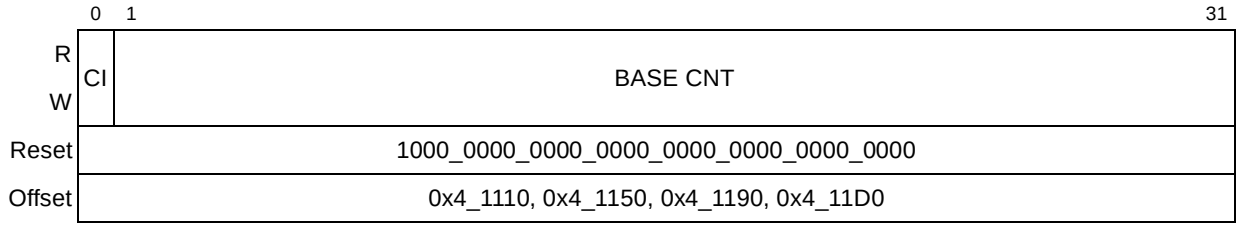


Figure 10-11. Global Timer Base Count Registers (GTBCR*n*)

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.

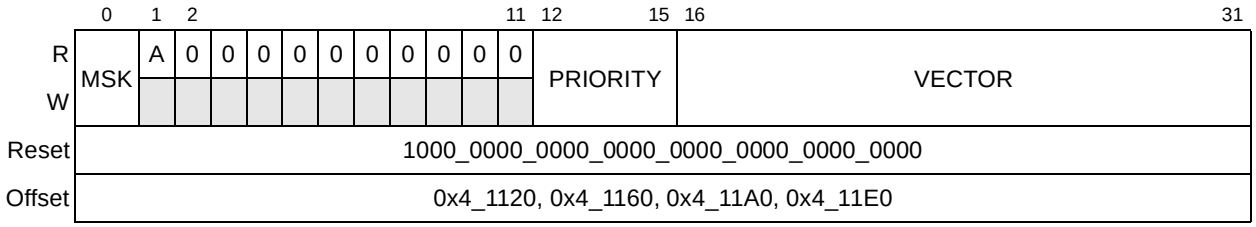


Figure 10-12. Global Timer Vector/Priority Registers (GTVPR*n*)

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.
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.

	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	P0
W																													
Reset	0000_0000_0000_0000_0000_0000_0000_0001																												
Offset	0x4_1130, 0x4_1170, 0x4_11B0, 0x4_11F0																												

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 MPC8540 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

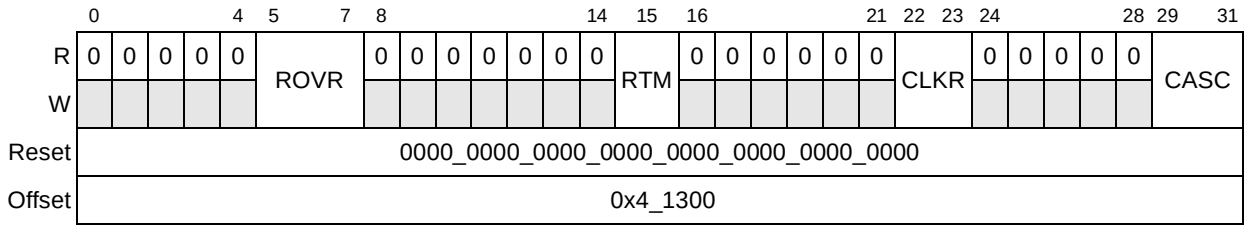


Figure 10-15. Timer Control Register (TCR)

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 20IRQ0 21IRQ1 . . . . 31IRQ11 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.

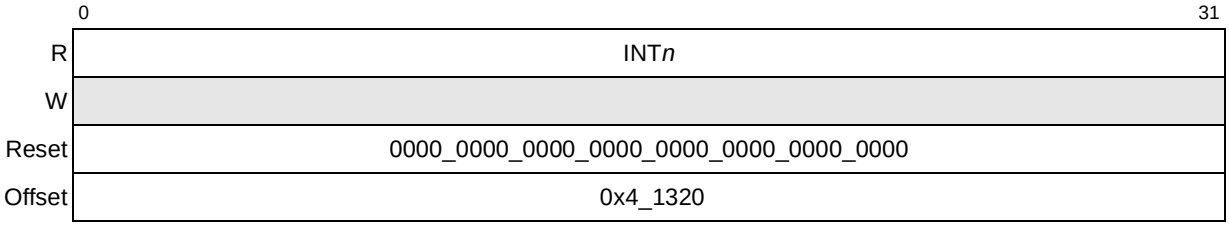


Figure 10-17. $\overline{\text{IRQ\_OUT}}$ Summary Register 1 (IRQSR1)

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.

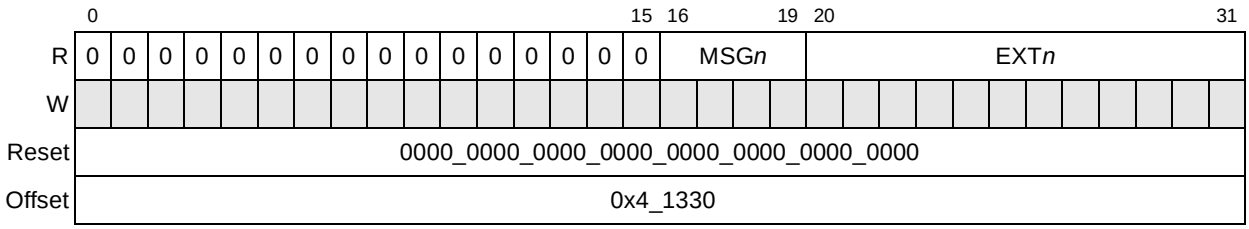


Figure 10-18. Critical Interrupt Summary Register 0 (CISR0)

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.

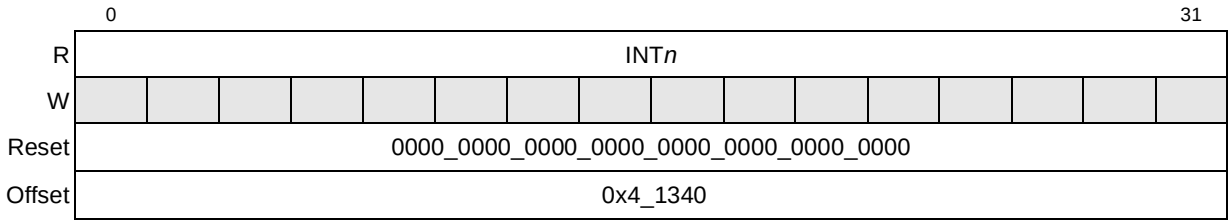


Figure 10-19. Critical Interrupt Summary Register 1 (CISR1)

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 19.3.2.2, “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.

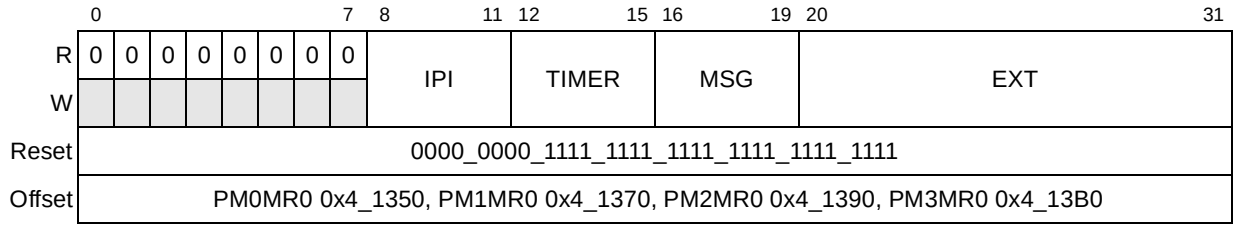


Figure 10-20. Performance Monitor Mask Registers (PM*n*MR0)

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.

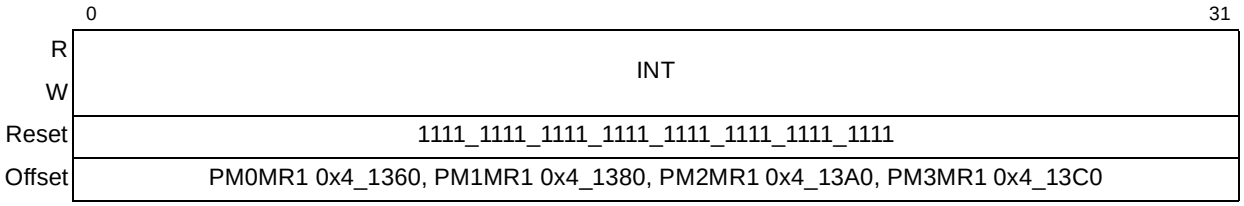


Figure 10-21. Performance Monitor Mask Registers (PM*n*MR1)

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.

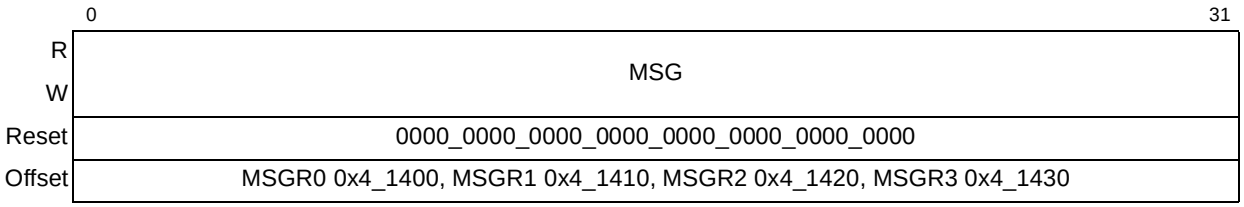


Figure 10-22. Message Registers (MSGRs)

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.

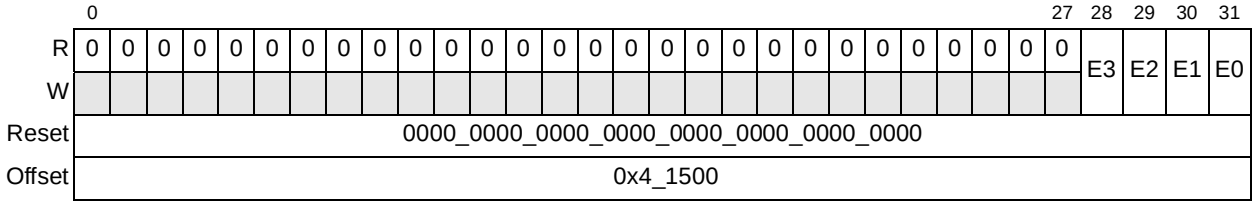


Figure 10-23. Message Enable Register (MER)

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.

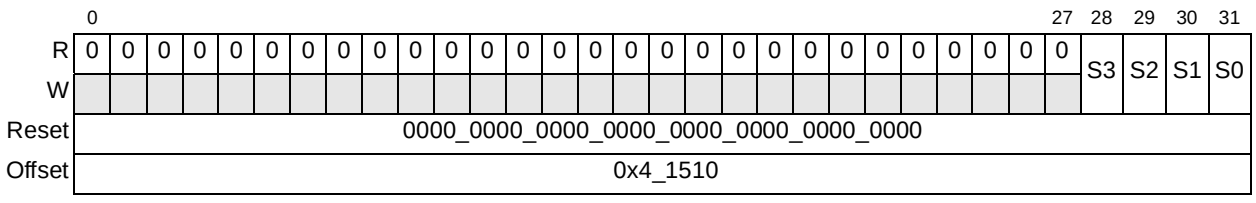


Figure 10-24. Message Status Register (MSR)

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.

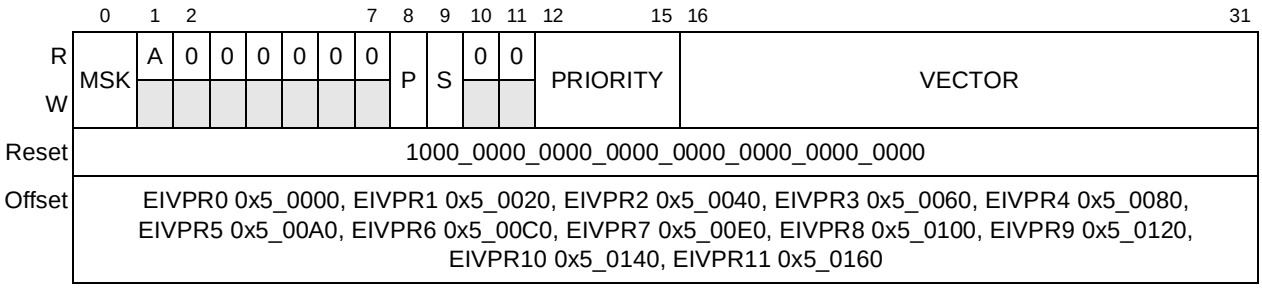


Figure 10-25. External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)

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	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 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 MPC8540 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	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 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 MPC8540 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 MPC8540 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 MPC8540 has only one core, there is only one set of per-CPU registers, each register having two addresses. For example, the CTPR is located normally at 0x6_0080 and 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.

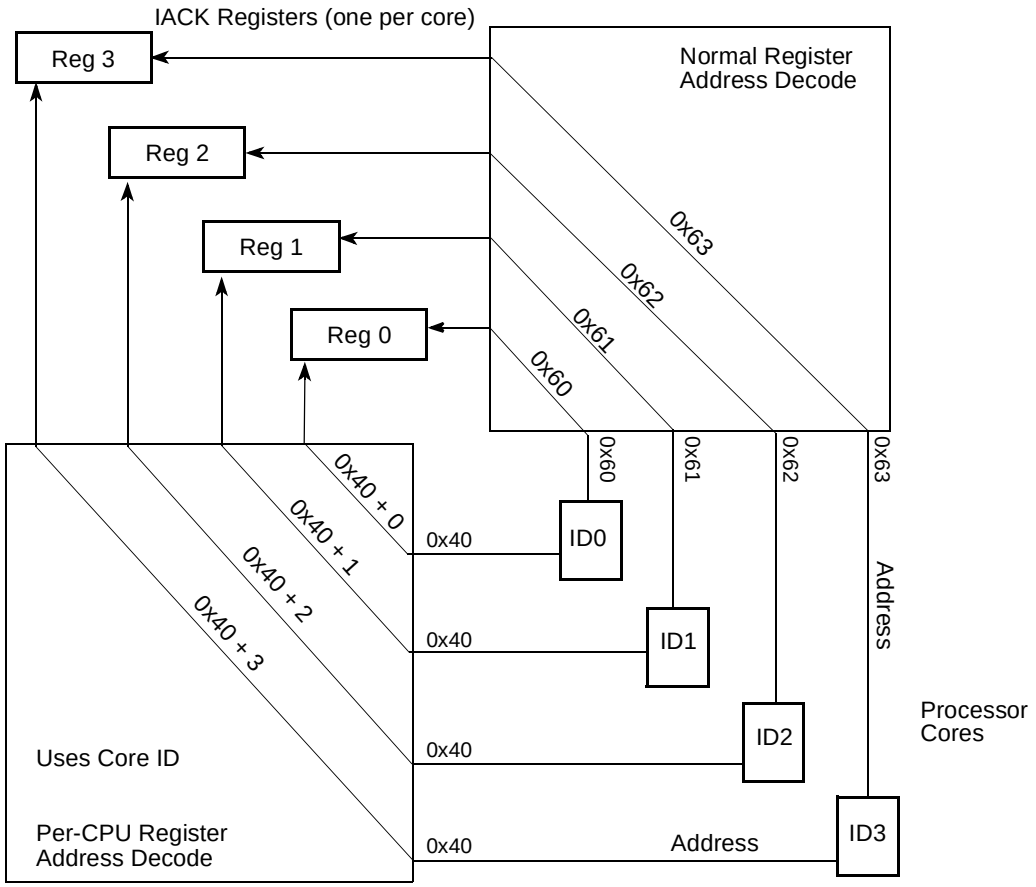


Figure 10-31. Per-CPU Register Address Decoding in a Four-Core Device

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 (MPC8540) 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.

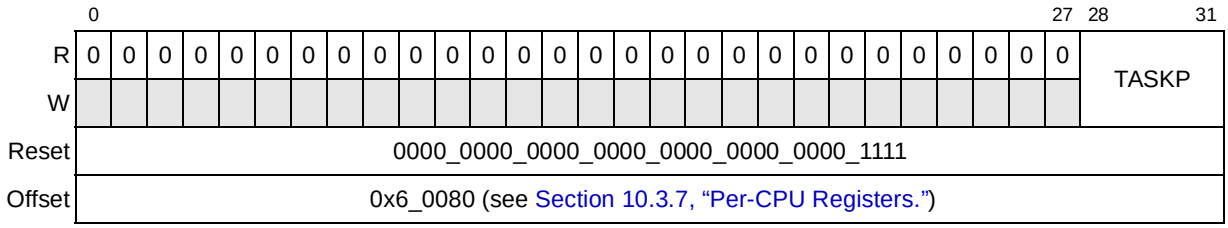


Figure 10-33. Processor Current Task Priority Register (CTPR)

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.

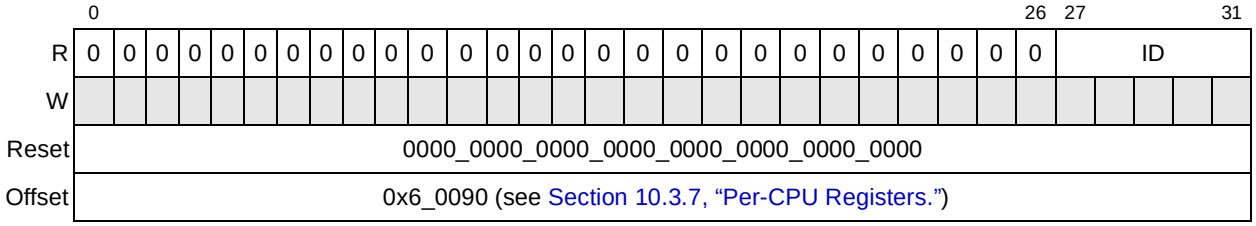


Figure 10-34. Processor Who Am I Register (WHOAMI)

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. 00000Processor 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).”

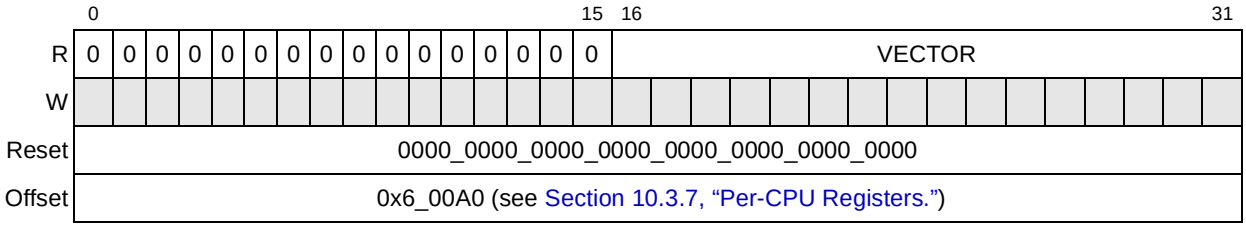


Figure 10-35. Processor Interrupt Acknowledge Register (IACK)

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.

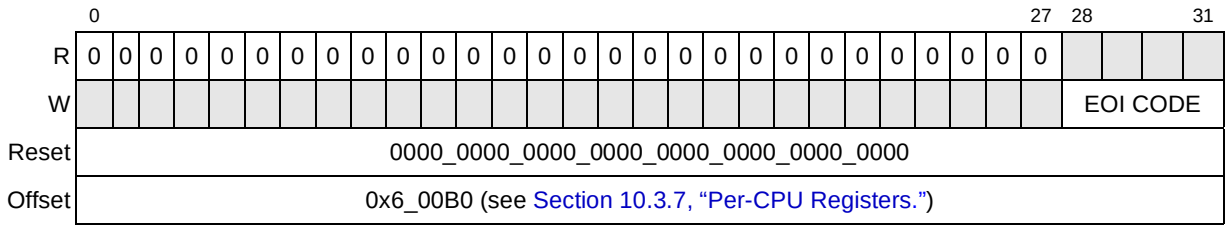


Figure 10-36. End of Interrupt Register (EOI)

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)

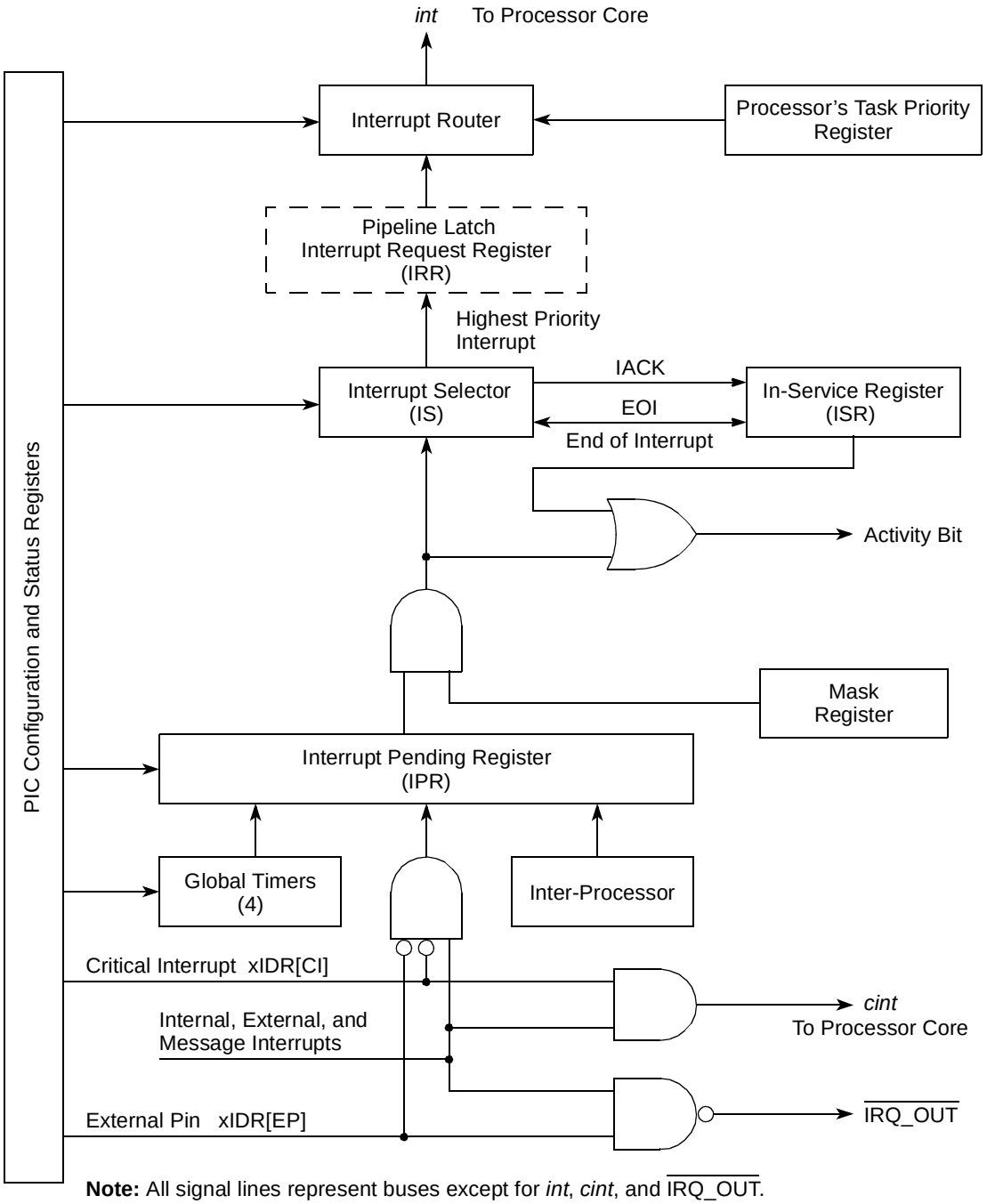
10.4 Functional Description

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:

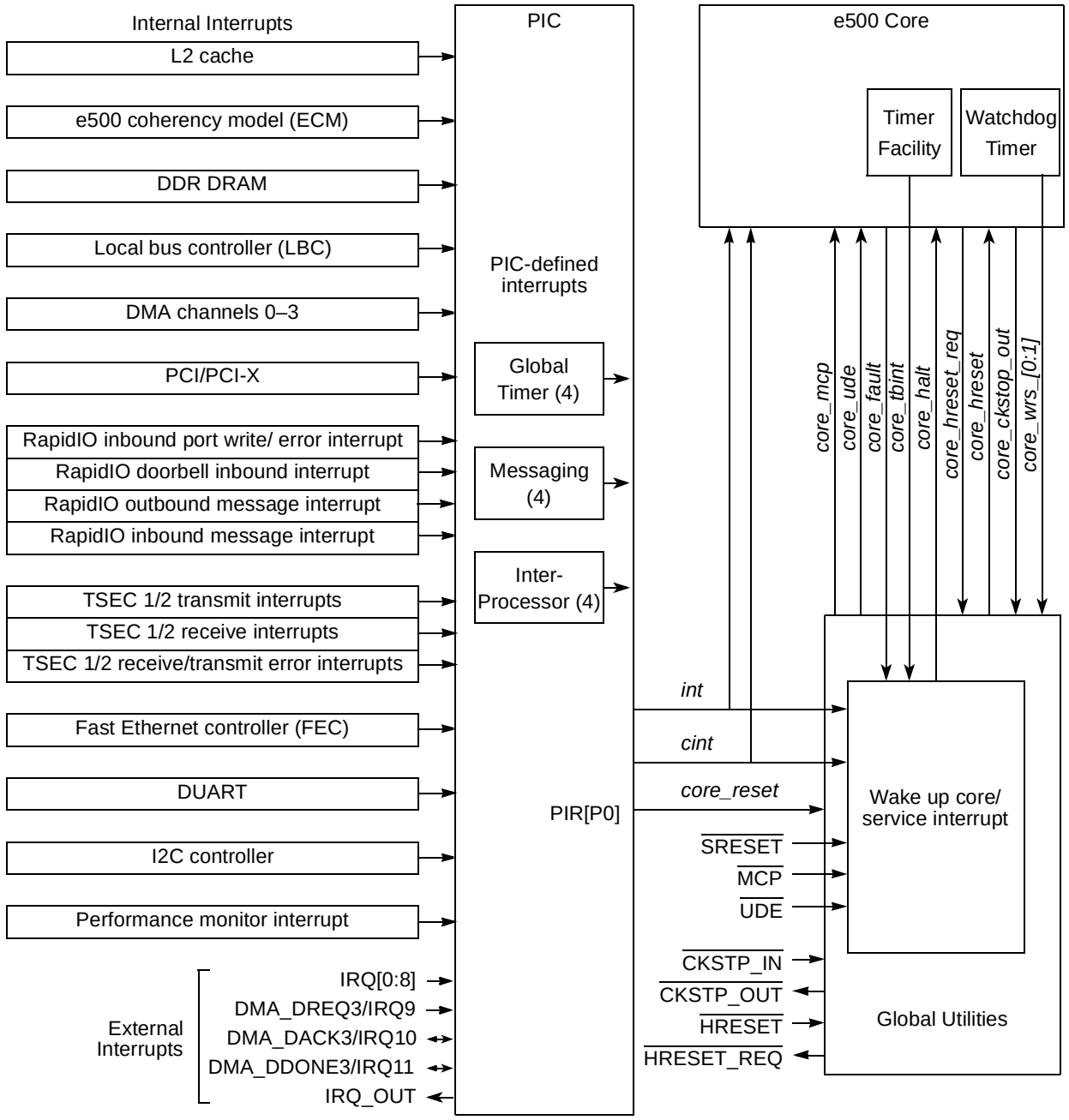
Because the default polarity/sense for external interrupts is edge-sensitive, and edge-sensitive interrupts are not cleared until they are acknowledged, it is possible for the PIC to store spurious edges detected during power-up as pending external interrupts. If software permanently configures an external 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.

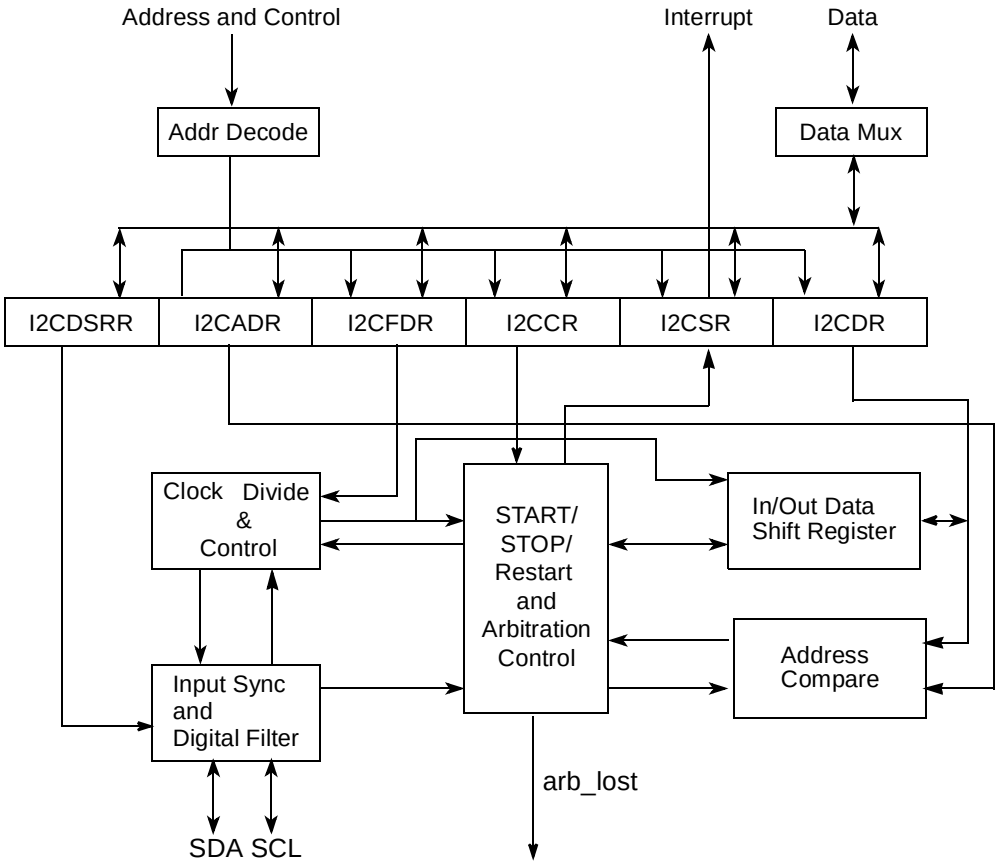


Figure 11-1. I²C Block Diagram

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 MPC8540 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 *MPC8540 Integrated Host 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 MPC8540 is programmed as an I2C slave. SCL also performs as an output when the MPC8540 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 MPC8540 is in a receiving mode. SDA also performs as an output signal when the MPC8540 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.

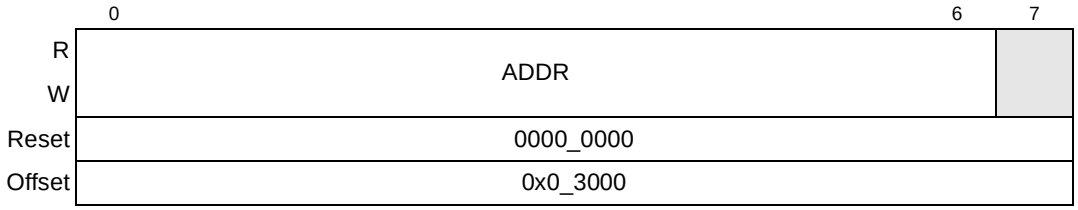


Figure 11-2. I²C Address Register (I2CADR)

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.

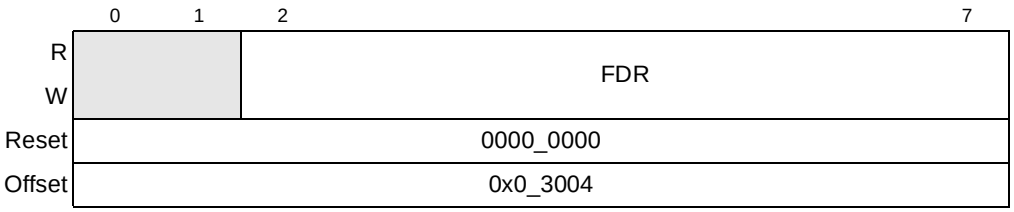


Figure 11-3. I²C Frequency Divider Register (I2CFDR)

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.

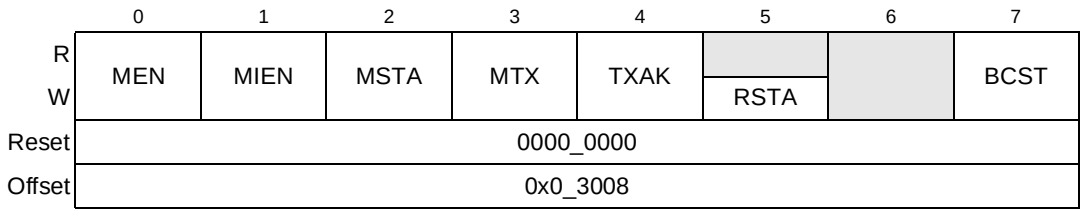


Figure 11-4. I²C Control Register (I2CCR)

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.

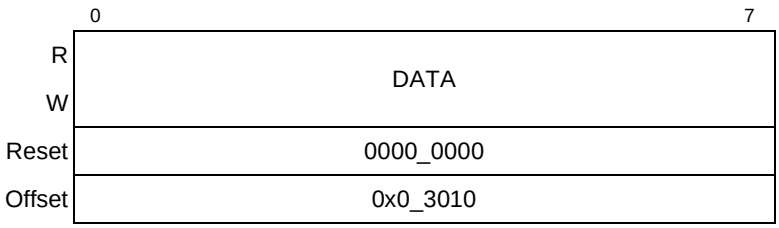


Figure 11-6. I²C Data Register (I2CDR)

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.

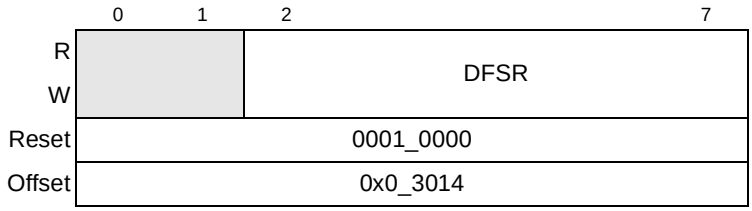


Figure 11-7. I2C Digital Filter Sampling Rate Register (I2CDFSRR)

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.

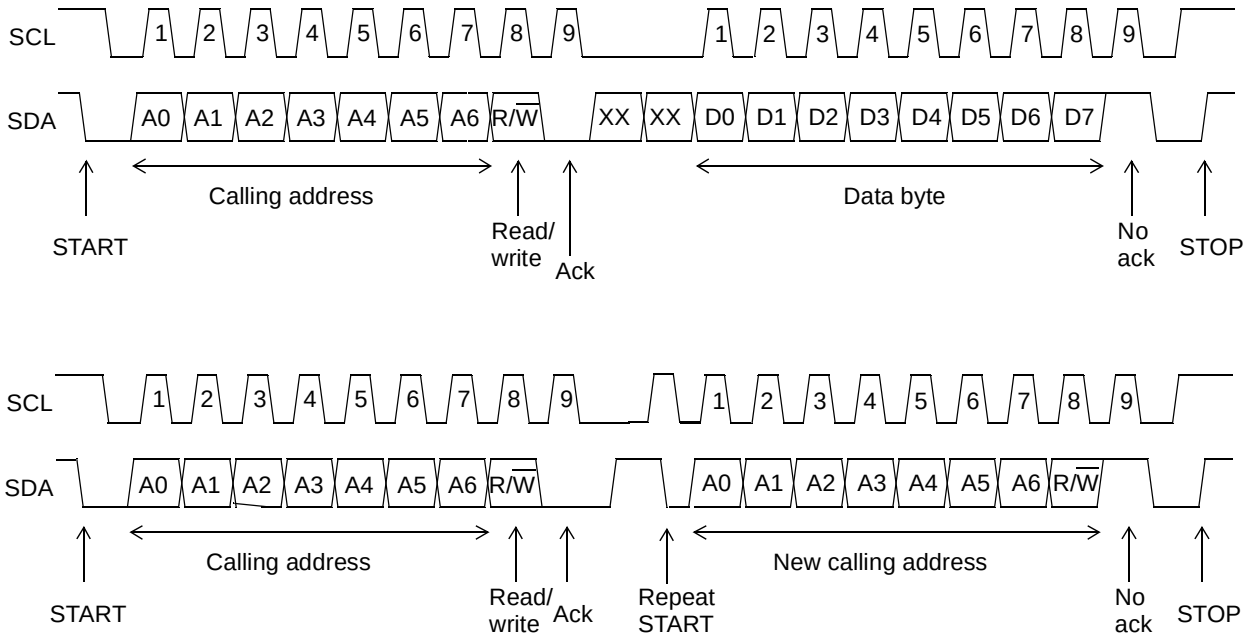


Figure 11-8. I²C Interface Transaction Protocol

11.4.1.1 START Condition

When the I²C bus is not engaged (both SDA and SCL lines are at logic high), a master can initiate a transfer by sending a START condition. As shown in Figure 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 I²C module is operating as a master, it must not transmit an address that is the same as its slave address. An I²C device cannot be master and slave at the same time.

Only the slave with a calling address that matches the one transmitted by the master responds by returning an acknowledge bit (pulling the SDA signal low at the 9th clock) as shown in Figure 11-8. If no slave acknowledges the address, the master should generate a STOP condition or a repeated START condition.

When slave addressing is successful (and SCL returns to zero), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/$\overline{W}$ bit sent by the calling master.

The 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 I^2C interface is a true multiple-master bus that allows more than one master device to be connected on it. If two or more masters simultaneously try to control the bus, each master's clock synchronization procedure (including the I^2C module) determines the bus clock—the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. A bus master loses arbitration if it transmits a logic 1 on SDA while another master transmits a logic 0. The losing masters immediately switch to slave-receive mode and stop driving the SDA line. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, the I^2C unit sets the I2CSR[MAL] status bit to indicate the loss of arbitration and, as a slave, services the transaction if it is directed to itself.

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

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

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 4.4.3.6, “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 4.4.3.6, “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 4.4.3.6, “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 MPC8540 uses 0b101_0000 for the EEPROM calling address. The first EEPROM to be addressed must be programmed to respond to this address, or an error is generated. If more 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.

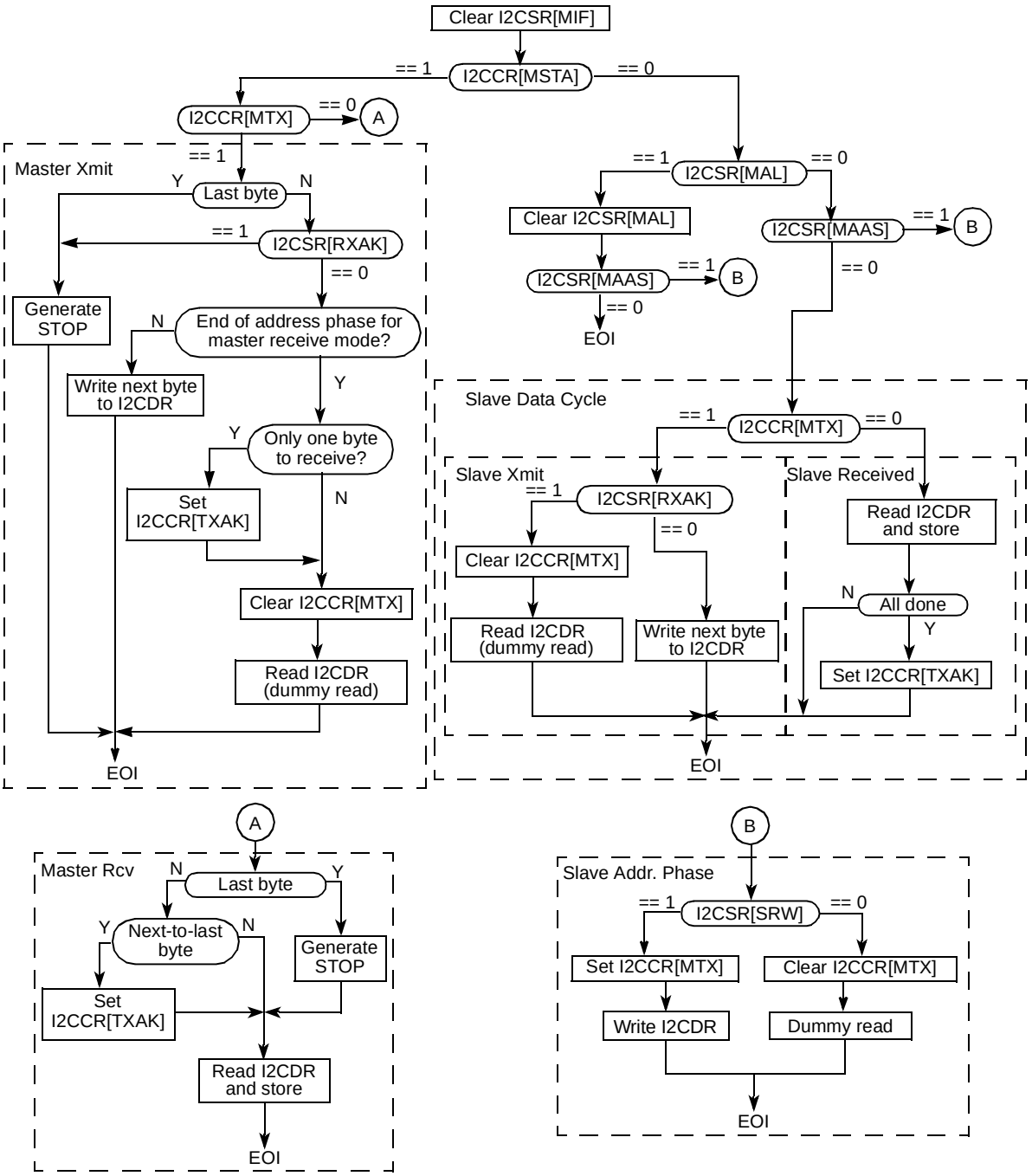


Figure 11-11. Example I2C Interrupt Service Routine Flowchart

Chapter 12
DUART

This chapter describes the dual universal asynchronous receiver/transmitters (DUART) of the MPC8540. It describes the functional operation, the DUART initialization sequence, and the programming details for the DUART registers and features.

12.1 Overview

The DUART of the MPC8540 consists of two universal asynchronous receiver/transmitters (UARTs). The UARTs act independently; all references to UART refer to one of these receiver/transmitters. Each UART is clocked by the core complex bus (CCB) clock. The DUART programming model is compatible with the PC16552D.

The UART interface is point to point, meaning that only two UART devices are attached to the connecting signals. As shown in Figure 12-1, each UART module consists of the following:

- Receive and transmit buffers
- Clear to send ($\overline{\text{CTS}}$) input port and request to send ($\overline{\text{RTS}}$) output port for data flow control
- 16-bit counter for baud rate generation
- Interrupt control logic

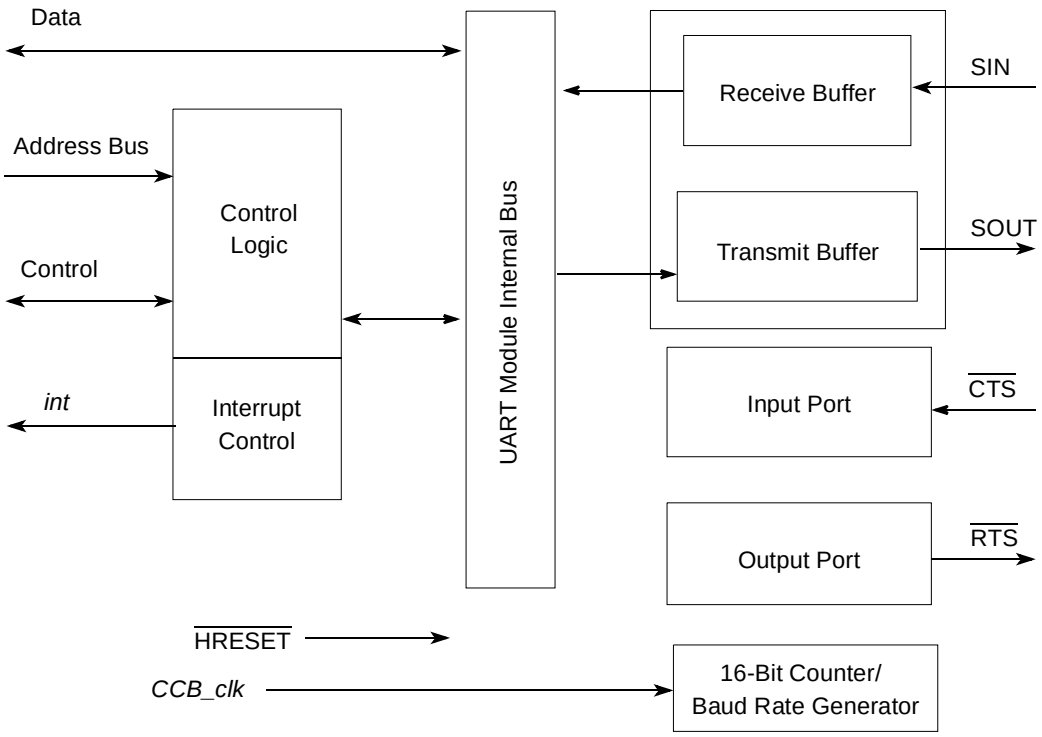


Figure 12-1. UART Block Diagram

12.1.1 Features

The DUART includes these distinctive features:

- Full-duplex operation
- Programming model compatible with the original PC16450 UART and the PC16550D (an improved version of the PC16450 that also operates in FIFO mode)
- PC16450 register reset values
- FIFO mode for both transmitter and receiver, providing 16-byte FIFOs
- Serial data encapsulation and decapsulation with standard asynchronous communication bits (START, STOP, and parity)
- Maskable transmit, receive, line status, and MODEM status interrupts
- Software-programmable baud generators that divide the CCB clock by 1 to ($2^{16} – 1$) and generate a 16x clock for the transmitter and receiver engines
- Clear to send ($\overline{\text{CTS}}$) and ready to send ($\overline{\text{RTS}}$) MODEM control functions
- Software-selectable serial interface data format (data length, parity, 1/1.5/2 STOP bit, baud rate)
- Line and MODEM status registers
- Line-break detection and generation

- Internal diagnostic support, local loopback, and break functions
- Prioritized interrupt reporting
- Overrun, parity, and framing error detection

12.1.2 Modes of Operation

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the CCB clock.

The transmitter accepts parallel data from a write to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register of the transmitter FIFO. The transmitter converts the data to a serial bit stream inserting the appropriate start, stop, and optional parity bits. Finally, it outputs a composite serial data stream on the channel transmitter serial data output signal (SOUT). The transmitter status may be polled or interrupt driven.

The receiver accepts serial data bits on the channel receiver serial data input signal (SIN), converts it to parallel format, checks for a start bit, parity (if any), stop bits, and transfers the assembled character (with start, stop, parity bits removed) from the receiver buffer (or FIFO) in response to a read of the UART's receiver buffer register (URBR). The receiver status may be polled or interrupt driven.

12.2 External Signal Descriptions

This section contains a signal overview and detailed signal descriptions.

12.2.1 Signal Overview

Table 12-1 summarizes the DUART signals. Note that although the actual device signal names are prepended with the UART_ prefix as shown in the table, the functional (abbreviated) signal names are often used throughout this chapter.

Table 12-1. DUART Signal Overview

Signal Name	I/O	Pins	Reset Value	State Meaning
UART_SIN[0:1]	I	2	1	Serial in data UART0 and UART1
UART_SOUT[0:1]	O	2	1	Serial out data UART0 and UART1
$\overline{\text{UART\_CTS}}$[0:1]	I	2	1	Clear to send UART0 and UART1
$\overline{\text{UART\_RTS}}$[0:1]	O	2	1	Request to send UART0 and UART1

12.2.2 Detailed Signal Descriptions

The eight DUART signals are described in detail in Table 12-2.

Table 12-2. DUART Signals—Detailed Signal Descriptions

Signal	I/O	Description	
UART_SIN[0:1]	I	Serial data in. Data is received on the receivers of UART0 and UART1 through its respective serial data input signal, with the least-significant bit received first.	
		State Meaning	Asserted/Negated—Represents the data being received on the UART interface.
		Timing	Assertion/Negation—An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to sample the data on SIN.
UART_SOUT[0:1]	O	Serial data out. The serial data output signals for the UART0 and UART1 are set ('mark' condition) when the transmitter is disabled, idle, or operating in the local loop-back mode. Data is shifted out on these signals, with the least significant bit transmitted first.	
		State Meaning	Asserted/Negated—Represents the data being transmitted on the respective UART interface.
		Timing	Assertion/Negation— An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to update and drive the data on SOUT.
$\overline{\text{UART\_CTS}}$[0:1]	I	Clear to send. These active-low inputs are the clear-to-send inputs. They are connected to the respective $\overline{\text{RTS}}$ outputs of the other UART devices on the bus. They can be programmed to generate an interrupt on change-of-state of the signal.	
		State Meaning	Asserted/Negated—Represent the clear to send condition for their respective UART.
		Timing	Assertion/Negation—Sampled at the rising edge of every CCB clock.
$\overline{\text{UART\_RTS}}$[0:1]	O	Request to send. $\overline{\text{UART_RTS}}$x are active-low output signals that can be programmed to be automatically negated and asserted by either the receiver or transmitter. When connected to the clear-to-send ($\overline{\text{CTS}}$) input of a transmitter, this signal can be used to control serial data flow.	
		State Meaning	Asserted/Negated—Represents the data being transmitted on the respective UART interface.
		Timing	Assertion/Negation—Updated and driven at the rising edge of every CCB clock.

12.3 Memory Map/Register Definition

There are two complete sets of DUART registers (one for UART0 and one for UART1). The two UARTs on the device are identical, except that the registers for UART0 are located at offsets 0x0_4500 (local), and the registers for UART1 are located at offsets 0x0_4600 (local). Throughout this chapter, the registers are described by a singular acronym: for example, LCR represents the line control register for either UART0 or UART1.

The 14 registers in each UART interface are used for configuration, control, and status. The divisor latch access bit, ULCR[DLAB], is used to access the divisor latch least- and most-significant bit registers and the alternate function register. Refer to Section 12.3.1.7, “Line Control Registers (ULCR0, ULCR1),” for more information on ULCR[DLAB].

All the DUART registers are one byte wide. Reads and writes to these registers must be byte-wide operations. Table 12-3 provides a register summary with references to the section and page that

contains detailed information about each register. Undefined byte address spaces within offset 0x000–0xFFF are reserved.

Table 12-3. DUART Register Summary

Offset	Register	Access	Reset	Section/Page
0x0_4500	URBR—ULCR[DLAB] = 0 UART0 receiver buffer register	R	0x00	12.3.1.1/12-6
0x0_4500	UTHR—ULCR[DLAB] = 0 UART0 transmitter holding register	W	0x00	12.3.1.2/12-6
0x0_4500	UDLB—ULCR[DLAB] = 1 UART0 divisor least significant byte register	R/W	0x00	12.3.1.3/12-7
0x0_4501	UIER—ULCR[DLAB] = 0 UART0 interrupt enable register	R/W	0x00	12.3.1.4/12-9
0x0_4501	UDMB—ULCR[DLAB] = 1 UART0 divisor most significant byte register	R/W	0x00	12.3.1.3/12-7
0x0_4502	UIIR—ULCR[DLAB] = 0 UART0 interrupt ID register	R	0x01	12.3.1.5/12-10
0x0_4502	UFCR—ULCR[DLAB] = 0 UART0 FIFO control register	W	0x00	12.3.1.6/12-11
0x0_4502	UAFR—ULCR[DLAB] = 1 UART0 alternate function register	R/W	0x00	12.3.1.12/12-17
0x0_4503	ULCR—ULCR[DLAB] = x UART0 line control register	R/W	0x00	12.3.1.7/12-12
0x0_4504	UMCR—ULCR[DLAB] = x UART0 MODEM control register	R/W	0x00	12.3.1.8/12-14
0x0_4505	ULSR—ULCR[DLAB] = x UART0 line status register	R	0x60	12.3.1.9/12-15
0x0_4506	UMSR—ULCR[DLAB] = x UART0 MODEM status register	R	0x00	12.3.1.10/12-16
0x0_4507	USCR—ULCR[DLAB] = x UART0 scratch register	R/W	0x00	12.3.1.11/12-17
0x0_4510	UDSR—ULCR[DLAB] = x UART0 DMA status register	R	0x01	12.3.1.13/12-18
0x0_4600	URBR—ULCR[DLAB] = 0 UART1 receiver buffer register	R	0x00	12.3.1.1/12-6
0x0_4600	UTHR—ULCR[DLAB] = 0 UART1 transmitter holding register	W	0x00	12.3.1.2/12-6
0x0_4600	UDLB—ULCR[DLAB] = 1 UART1 divisor least significant byte register	R/W	0x00	12.3.1.3/12-7
0x0_4601	UIER—ULCR[DLAB] = 0 UART1 interrupt enable register	R/W	0x00	12.3.1.4/12-9
0x0_4601	UDMB_ULCR[DLAB] = 1 UART1 divisor most significant byte register	R/W	0x00	12.3.1.3/12-7
0x0_4602	UIIR—ULCR[DLAB] = 0 UART1 interrupt ID register	R	0x01	12.3.1.5/12-10
0x0_4602	UFCR—ULCR[DLAB] = 0 UART1 FIFO control register	W	0x00	12.3.1.6/12-11
0x0_4602	UAFR—ULCR[DLAB] = 1 UART1 alternate function register	R/W	0x00	12.3.1.12/12-17
0x0_4603	ULCR—ULCR[DLAB] = x UART1 line control register	R/W	0x00	12.3.1.7/12-12
0x0_4604	UMCR—ULCR[DLAB] = x UART1 MODEM control register	R/W	0x00	12.3.1.8/12-14
0x0_4605	ULSR—ULCR[DLAB] = x UART1 line status register	R	0x60	12.3.1.9/12-15
0x0_4606	UMSR—ULCR[DLAB] = x UART1 MODEM status register	R	0x00	12.3.1.10/12-16
0x0_4607	USCR—ULCR[DLAB] = x UART1 scratch register	R/W	0x00	12.3.1.11/12-17
0x0_4610	UDSR—ULCR[DLAB] = x UART1 DMA status register	R	0x01	12.3.1.13/12-18

12.3.1 Register Descriptions

The following sections describe the MPC8540 UART0 and UART1 registers.

12.3.1.1 Receiver Buffer Registers (URBR0, URBR1)

These registers contain the data received from the transmitter on the UART buses. In FIFO mode, when read, they return the first byte received. For FIFO status information, refer to the UDSR[RXRDY] description.

Except for the case when there is an overrun, URBR returns the data in the order it was received from the transmitter. Refer to the ULSR[OE] description, Section 12.3.1.9, “Line Status Registers (ULSR0, ULSR1).” Figure 12-3 shows the receiver buffer registers. Note that these registers have same offset as the UTHRs.

Figure 12-2 shows the bits in the URBRs.

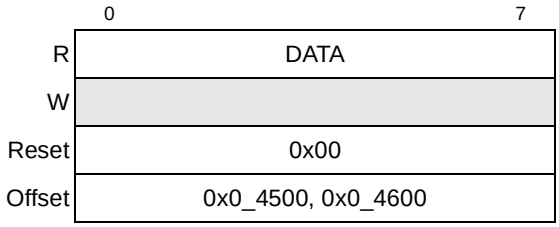


Figure 12-2. Receiver Buffer Registers (URBR0, URBR1)

Table 12-4 describes the fields of URBR.

Table 12-4. URBR Field Definition

Bits	Name	Description
0–7	DATA	Data received from the transmitter on the UART bus (read only)

12.3.1.2 Transmitter Holding Registers (UTHR0, UTHR1)

A write to these 8-bit registers causes the UART devices to transfer 5–8 data bits on the UART bus in the format set up in the ULCR (line control register). In FIFO mode, data written to UTHR is placed into the FIFO. The data written to UTHR is the data sent onto the UART bus, and the first byte written to UTHR will be the first byte onto the bus. UDSR[$\overline{\text{TXRDY}}$] indicates when the FIFO is full. Refer to the Table 12-21 and the Table 12-22 for more details.

Figure 12-3 shows the bits in the UTHRs.

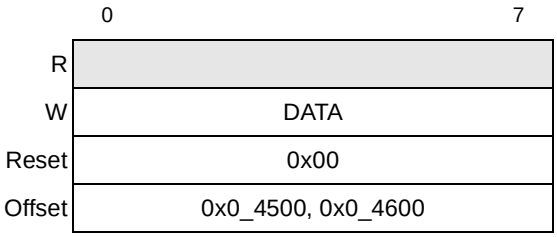


Figure 12-3. Transmitter Holding Registers (UTHR0, UTHR1)

Table 12-5 describes the fields of UTHR.

Table 12-5. UTHR Field Definition

Bits	Name	Description
0–7	DATA	Data that is written to UTHR (write only)

12.3.1.3 Divisor Most and Least Significant Byte Registers (UDMB and UDLB)

The divisor least significant byte register (UDLB) is concatenated with the divisor most significant byte register (UDMB) to create the divisor used to divide the input clock into the DUART. The output frequency of the baud generator is 16 times the baud rate; therefore the desired baud rate = platform clock frequency / (16 × [UDMB||UDLB]). Equivalently, þþ [UDMB||UDLB:0b0000]þ = platform clock frequency / desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in Table 12-8.

Figure 12-4 shows the bits in the UDMBs.

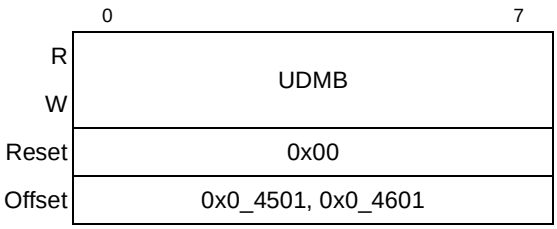


Figure 12-4. Divisor Most Significant Byte Registers (UDMB0, UDMB1)

Table 12-6 describes the fields of UDMB registers.

Table 12-6. UDMB Field Definition

Bits	Name	Description
0–7	UDMB	Divisor most-significant byte

Figure 12-5 shows the bits in the UDLBs.

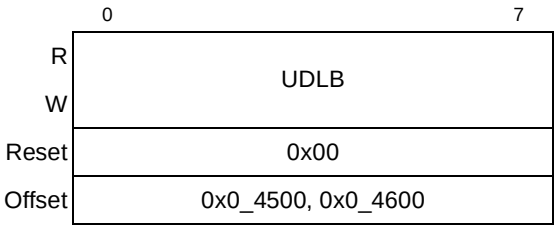


Figure 12-5. Divisor Least Significant Byte Registers (UDLB0, UDLB1)

Table 12-7 describes the fields of UDLB registers.

Table 12-7. UDLB Field Definition

Bits	Name	Description
0–7	UDLB	Divisor least-significant byte. This is concatenated with UDMB.

Table 12-8 shows baud rate when the input clock is at certain frequencies.

Table 12-8. Baud Rate Examples

Baud Rate (Decimal)	Divisor		Input Clock(CCB) Frequency (MHz)	Percent Error (Decimal)
	Decimal	Hex		
9,600	1736	6C8	266	0.0064
19,200	868	364	266	0.0064
38,400	434	1B2	266	0.0064
56,000	298	12A	266	0.1280
128,000	130	82	266	0.1600
256,000	65	41	266	0.1600
9,600	2170	87A	333	0.0064
19,200	1085	43D	333	0.0064
38,400	543	21F	333	0.0858
56,000	372	174	333	0.0064
128,000	163	A3	333	0.1472
256,000	81	51	333	0.4672

To get the percent error value, the following three steps are taken:

1. The input clock frequency (ICF) is divided by the actual frequency input (AFI) to get the correct divisor value (ICF/AFI where AFI = baud rate × 16).
2. The divisor value is subtracted from 1.

3. The result from step two is multiplied by 100 to calculate the final percent error. The result is calculated in absolute value (no negative numbers).

These steps can be described with the following equation:

Percent error value = (1 – AFI/ICF) × 100

12.3.1.4 Interrupt Enable Register (UIER)

The UIER gives the user the ability to mask specific UART interrupts to the MPC8540 programmable interrupt controller (PIC).

Figure 12-6 shows the bits in the UIER.

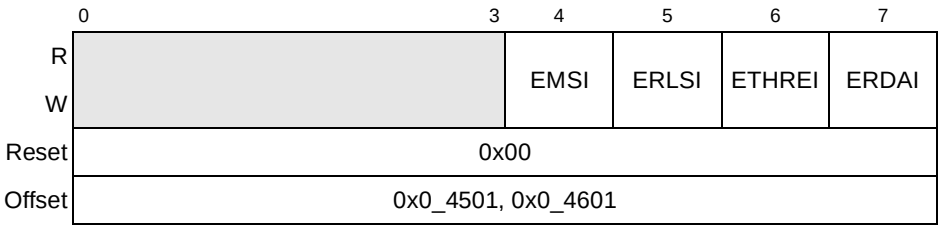


Figure 12-6. Interrupt Enable Register (UIER)

Table 12-9 describes the fields of UIER.

Table 12-9. UIER Field Definitions

Bits	Name	Description
0–3	—	Reserved
4	EMSI	Enable MODEM status interrupt 0 Mask interrupts caused by UMSR[DCTS] being set 1 Enable and assert interrupts when the clear-to-send bit in the UART MODEM status register (UMSR) changes state
5	ERLSI	Enable receiver line status interrupt 0 Mask interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set 1 Enable and assert interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set
6	ETHREI	Enable transmitter holding register empty interrupt 0 Mask interrupt when ULSR[THRE] is set 1 Enable and assert interrupts when ULSR[THRE] is set
7	ERDAI	Enable received data available interrupt 0 Mask interrupt when new receive data is available or receive data time out has occurred 1 Enable and assert interrupts when a new data character is received from the external device and/or a time-out interrupt occurs in the FIFO mode

12.3.1.5 Interrupt ID Registers (UIIR0, UIIR1)

The UIIRs indicate when an interrupt is pending from the corresponding UART and what type of interrupt is active. They also indicate if the FIFOs are enabled.

The DUART prioritizes interrupts into four levels and records these in the corresponding UIIR. The four levels of interrupt conditions in order of priority are:

1. Receiver line status
2. Received data ready/character time-out
3. Transmitter holding register empty
4. MODEM status

See Table 12-11 for more details.

When the UIIR is read, the associated DUART serial channel freezes all interrupts and indicates the highest priority pending interrupt. While this read transaction is occurring, the associated DUART serial channel records new interrupts, but does not change the contents of UIIR until the read access is complete.

Figure 12-7 shows the bits in the UIIR.

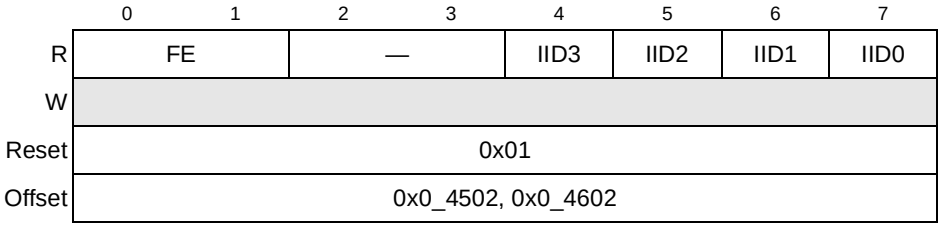


Figure 12-7. Interrupt ID Registers (UIIR)

Table 12-10 describes the fields of the UIIR.

Table 12-10. UIIR Field Definitions

Bits	Name	Description
0–1	FE	FIFOs enabled. Reflects the setting of UFCR[FEN]
2–3	—	Reserved
4	IID3	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in Table 12-11. IID3 is set along with IID2 only when a timeout interrupt is pending for FIFO mode.
5–6	IID2–1	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in Table 12-11.
7	IID0	IID0 indicates when an interrupt is pending. 0 The UART has an active interrupt ready to be serviced. 1 No interrupt is pending.

The bits contained in the UIIR registers are described in Table 12-11.

Table 12-11. UIIR IID Bits Summary

IID Bits IID[3–0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0b0001	—	—	—	—
0b0110	Highest	Receiver line status	Overrun error, parity error, framing error, or break interrupt	Read the line status register.
0b0100	Second	Received data available	Receiver data available or trigger level reached in FIFO mode	Read the receiver buffer register or interrupt is automatically reset if the number of bytes in the receiver FIFO drops below the trigger level.
0b1100	Second	Character time-out	No characters have been removed from or input to the receiver FIFO during the last 4 character times and there is at least one character in the receiver FIFO during this time.	Read the receiver buffer register.
0b0010	Third	UTHR empty	Transmitter holding register is empty	Read the UIIR or write to the UTHR.
0b0000	Fourth	MODEM status	$\overline{\text{CTS}}$ input value changed since last read of UMSR	Read the UMSR.

12.3.1.6 FIFO Control Registers (UFCR0, UFCR1)

The UFCR, a write-only register, is used to enable and clear the receiver and transmitter FIFOs, set a receiver FIFO trigger level to control the received data available interrupt, and select the type of DMA signaling.

When the UFCR bits are written, the FIFO enable bit must also be set or else the UFCR bits are not programmed. When changing from FIFO mode to 16450 mode (non-FIFO mode) and vice versa, data is automatically cleared from the FIFOs.

After all the bytes in the receiver FIFO are cleared, the receiver internal shift register is not cleared. Similarly, the bytes are cleared in the transmitter FIFO, but the transmitter internal shift register is not cleared. Both TFR and RFR are self-clearing bits.

Figure 12-8 shows the bits in the UFCRs.

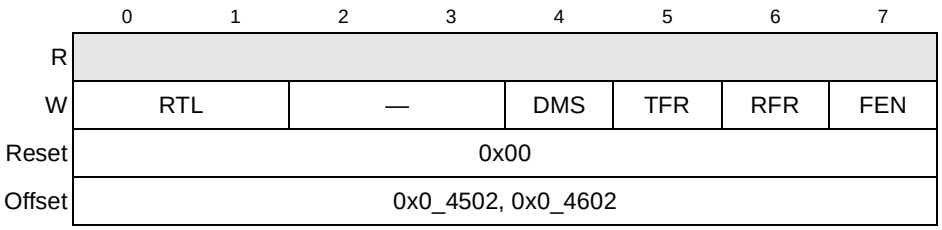


Figure 12-8. FIFO Control Registers (UFCR0, UFCR1)

Table 12-12 describes the fields of the UFCRs.

Table 12-12. UFCR Field Definitions

Bits	Name	Description
0–1	RTL	Receiver trigger level. A received data available interrupt occurs when UIER[ERDAI] is set and the number of bytes in the receiver FIFO equals the designated interrupt trigger level as follows: 00 1 byte 01 4 bytes 10 8 bytes 11 14 bytes
2–3	—	Reserved
4	DMS	DMA mode select. See Section 12.4.5.2, “DMA Mode Select,” for more information. 0 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 0. 1 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 1 if UFCR[FEN] = 1.
5	TFR	Transmitter FIFO reset 0 No action 1 Clears all bytes in the transmitter FIFO and resets the FIFO counter/pointer to 0
6	RFR	Receiver FIFO reset 0 No action 1 Clears all bytes in the receiver FIFO and resets the FIFO counter/pointer to 0
7	FEN	FIFO enable 0 FIFOs are disabled and cleared 1 Enables the transmitter and receiver FIFOs

12.3.1.7 Line Control Registers (ULCR0, ULCR1)

The ULCRs specify the data format for the UART bus and set the divisor latch access bit ULCR[DLAB], which controls the ability to access the divisor latch least and most significant bit registers and the alternate function register.

After initializing the ULCR, the software should not re-write the ULCR when valid transfers on the UART bus are active. The software should not re-write the ULCR until the last STOP bit has been received and there are no new characters being transferred on the bus.

The stick parity bit, ULCR[SP], assigns a set parity value for the parity bit time slot sent on the UART bus. The set value is defined as mark parity (logic 1) or space parity (logic 0). ULCR[PEN] and ULCR[EPS] help determine the set parity value. See Table 12-14 for more information. ULCR[NSTB], defines the number of STOP bits to be sent at the end of the data transfer. The receiver only checks the first STOP bit, regardless of the number of STOP bits selected. The word length select bits (1 and 0) define the number of data bits that are transmitted or received as a serial character. The word length does not include START, parity, and STOP bits.

Figure 12-9 shows the bits in the ULCRs.

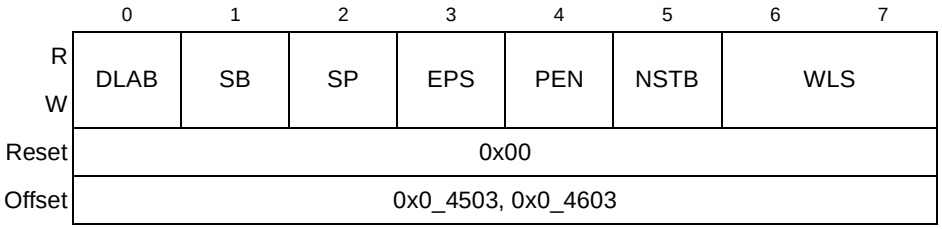


Figure 12-9. Line Control Register (ULCR)

Table 12-13 describes the fields of the ULCRs.

Table 12-13. ULCR Field Definitions

Bits	Name	Description
0	DLAB	Divisor latch access bit 0 Access to all registers except UDLB, UAFR, and UDMB 1 Ability to access divisor latch least and most significant byte registers and alternate function register (UAFR)
1	SB	Set break 0 Send normal UTHR data onto the serial output (SOUT) signal 1 Force logic 0 to be on the SOUT signal. Data in the UTHR is not affected
2	SP	Stick parity 0 Stick parity is disabled. 1 If PEN = 1 and EPS = 1, space parity is selected. And if PEN = 1 and EPS = 0, mark parity is selected.
3	EPS	Even parity select. See Table 12-14 for more information. 0 If PEN = 1 and SP = 0, odd parity is selected. 1 If PEN = 1 and SP = 0, even parity is selected.
4	PEN	Parity enable 0 No parity generation and checking 1 Generate parity bit as a transmitter, and check parity as a receiver
5	NTSB	Number of STOP bits 0 One STOP bit is generated in the transmitted data. 1 When a 5-bit data length is selected, 1 ½ STOP bits are generated. When either a 6-, 7-, or 8-bit word length is selected, two STOP bits are generated.
6–7	WLS	Word length select. Number of bits that comprise the character length. The word length select values are as follows: 00 5 bits 01 6 bits 10 7 bits 11 8 bits

Table 12-14. Parity Selection Using ULCR[PEN], ULCR[SP], and ULCR[EPS]

PEN	SP	EPS	Parity Selected
0	0	0	No parity
0	0	1	No parity
0	1	0	No parity
0	1	1	No parity
1	0	0	Odd parity
1	0	1	Even parity
1	1	0	Mark parity
1	1	1	Space parity

12.3.1.8 MODEM Control Registers (UMCR0, UMCR1)

The UMCRs control the interface with the external peripheral device on the UART bus.

Figure 12-10 shows the bits in the UMCRs.

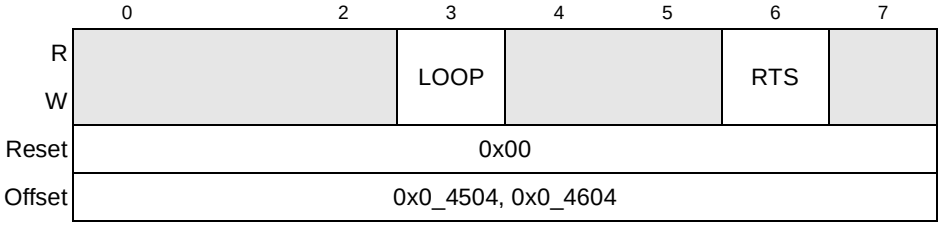


Figure 12-10. Modem Control Register (UMCR)

Table 12-15 describes the fields of UMCRs.

Table 12-15. UMCR Field Definitions

Bits	Name	Description
0–2	—	Reserved
3	LOOP	Local loop-back mode 0 Normal operation 1 Functionally, the data written to UTHR can be read from URBR of the same UART, and UMCR[RTS] is tied to UMSR[CTS].
4–5	—	Reserved
6	RTS	Ready to send 0 Negates corresponding $\overline{\text{UART\_RTS}}$ output 1 Assert corresponding $\overline{\text{UART\_RTS}}$ output. Informs external MODEM or peripheral that the UART is ready for sending/receiving data
7	—	Reserved

12.3.1.9 Line Status Registers (ULSR0, ULSR1)

The ULSRs are read-only registers that monitor the status of the data transfer on the UART buses. To isolate the status bits from the proper character received through the UART bus, software should read the ULSR and then the URBR.

Figure 12-11 shows the bits in the ULSRs.

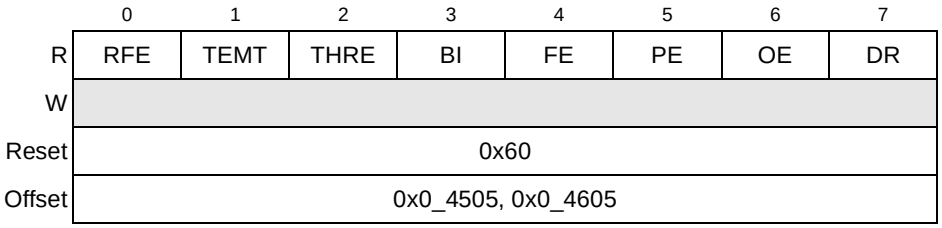


Figure 12-11. Line Status Register (ULSR)

Table 12-16 describes the fields of the ULSRs.

Table 12-16. ULSR Field Definitions

Bits	Name	Description
0	RFE	Receiver FIFO error 0 This bit is cleared when there are no errors in the receiver FIFO or on a read of the ULSR with no remaining receiver FIFO errors. 1 Set to one when one of the characters in the receiver FIFO encounters an error (framing, parity, or break interrupt)
1	TEMT	Transmitter empty 0 Either or both the UTHR or the internal transmitter shift register has a data character. In FIFO mode, a data character is in the transmitter FIFO or the internal transmitter shift register. 1 Both the UTHR and the internal transmitter shift register are empty. In FIFO mode, both the transmitter FIFO and the internal transmitter shift register are empty.
2	THRE	Transmitter holding register empty 0 The UTHR is not empty. 1 A data character has transferred from the UTHR into the internal transmitter shift register. In FIFO mode, the transmitter FIFO contains no data character.
3	BI	Break interrupt 0 This bit is cleared when the ULSR is read or when a valid data transfer is detected (that is, STOP bit is received). 1 Received data of logic 0 for more than START bit + Data bits + Parity bit + one STOP bits length of time. A break condition is expected to last at least two character lengths and a new character is not loaded until SIN returns to the mark state (logic 1) and a valid START is detected. In FIFO mode, a zero character is encountered in the FIFO (the zero character is at the top of the FIFO). In FIFO mode, only one zero character is stored. Note that the ULSR[BI] is set immediately after ULSR is read if bus remains zero and no mark state followed by a valid new character has been detected.

Table 12-16. ULSR Field Definitions (continued)

Bits	Name	Description
4	FE	Framing error 0 This bit is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register. 1 Invalid STOP bit for receive data (only the first STOP bit is checked). In FIFO mode, this bit is set when the character that detected a framing error is encountered in the FIFO (that is the character at the top of the FIFO). An attempt to resynchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit, so it assumes this logic 0 sample is a true START bit and then will receive the following new data.
5	PE	Parity error 0 This bit is cleared when ULSR is read or when a new character is loaded into the URBR. 1 Unexpected parity value encountered when receiving data. In FIFO mode, the character with the error is at the top of the FIFO.
6	OE	Overrun error 0 This bit is cleared when ULSR is read. 1 Before the URBR is read, the URBR was overwritten with a new character. The old character is loss. In FIFO mode, the receiver FIFO is full (regardless of the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. The old character was overwritten by the new character. Data in the receiver FIFO was not overwritten.
7	DR	Data ready 0 This bit is cleared when URBR is read or when all of the data in the receiver FIFO is read. 1 A character has been received in the URBR or the receiver FIFO.

12.3.1.10 MODEM Status Registers (UMSR0, UMSR1)

The UMSRs track the status of the MODEM (or external peripheral device) clear to send ($\overline{\text{CTS}}$) signal for the corresponding UART.

Figure 12-12 shows the bits in the UMSRs.

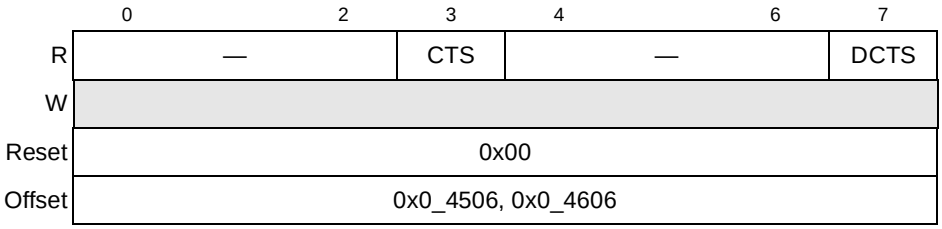


Figure 12-12. Modem Status Register (UMSR)

Table 12-17 describes the fields of the UMSRs.

Table 12-17. UMSR Field Definitions

Bits	Name	Description
0–2	—	Reserved
3	CTS	Clear to send. Represents the inverted value of the $\overline{\text{CTS}}$ input pin from the external peripheral device 0 Corresponding $\overline{\text{CTS}}n$ is negated 1 Corresponding $\overline{\text{CTS}}n$ is asserted. The MODEM or peripheral device is ready for data transfers.
4–6	—	Reserved
7	DCTS	Delta clear to send 0 No change on the corresponding $\overline{\text{CTS}}n$ signal since the last read of UMSR[CTS] 1 The $\overline{\text{CTS}}n$ value has changed, since the last read of UMSR[CTS]. Causes an interrupt if UIER[EMSI] is set to detect this condition

12.3.1.11 Scratch Registers (USCR0, USCR1)

The USCR registers are for debugging software or the DUART hardware. The USCRs do not affect the operation of the DUART.

Figure 12-13 shows the bits in USCRs.

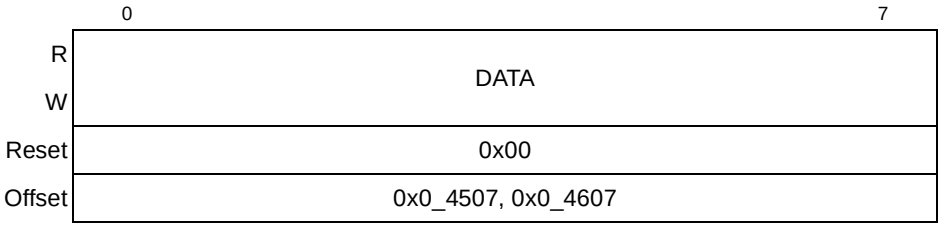


Figure 12-13. Scratch Register (USCR)

Table 12-18 describes the fields of the USCRs.

Table 12-18. USCR Field Definitions

Bits	Name	Description
0–7	DATA	Data

12.3.1.12 Alternate Function Registers (UAFR0, UAFR1)

The UAFRs give software the ability to write to both UART0 and UART1 registers simultaneously with the same write operation. The UAFRs also provide a means for the device's performance monitor to track the baud clock.

Figure 12-14 shows the bits in the UAFRs.

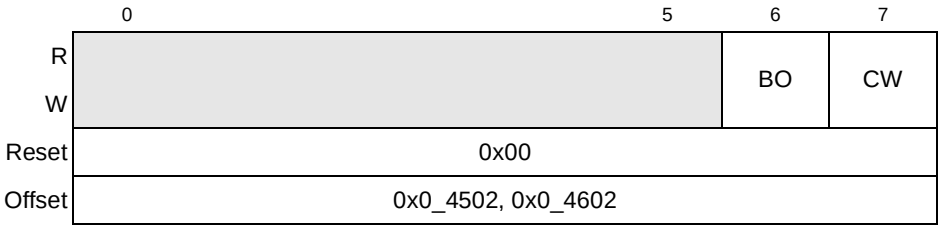


Figure 12-14. Alternate Function Register (UAFR)

Table 12-19 describes the fields of the UAFRs.

Table 12-19. UAFR Field Definitions

Bits	Name	Description
0–5	—	Reserved
6	BO	Baud clock select 0 The baud clock is not gated off. 1 The baud clock is gated off.
7	CW	Concurrent write enable 0 Disables writing to both UART0 and UART1 1 Enables concurrent writes to corresponding UART registers. A write to a register in UART0 is also a write to the corresponding register in UART1 and vice versa.

12.3.1.13 DMA Status Registers (UDSR0, UDSR1)

The DMA status registers (UDSRs) are read-only registers that return transmitter and receiver FIFO status. UDSRs also provide the ability to assist DMA data operations to and from the FIFOs.

Figure 12-15 shows the bits in UDSRs.

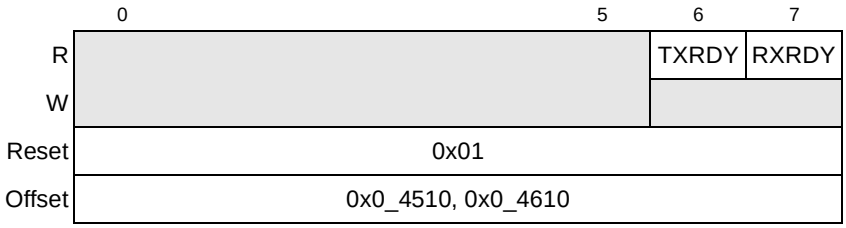


Figure 12-15. DMA Status Register (UDSR)

Table 12-20 describes the fields of the UDSRs.

Table 12-20. UDSR Field Definitions

Bits	Name	Description
0–5	—	Reserved
6	TXRDY	Transmitter ready. This read-only bit reflects the status of the transmitter FIFO or the UTHR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR. 0 The bit is cleared, as shown in Table 12-22. 1 This bit is set, as shown in Table 12-21.
7	RXRDY	Receiver ready. This read-only bit reflects the status of the receiver FIFO or URBR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR. 0 The bit is cleared, as shown in Table 12-24. 1 This bit is set, as shown in Table 12-23.

Table 12-21. UDSR[TXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is set after the first character is loaded into the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is set when the transmitter FIFO is full.

Table 12-22. UDSR[TXRDY] Cleared Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR. TXRDY remains clear when the transmitter FIFO is not yet full.

Table 12-23. UDSR[RXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is set when there are no characters in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is set when the trigger level has not been reached and there has been no time out.

Table 12-24. UDSR[RXRDY] Cleared

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is cleared when there is at least one character in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is cleared when the trigger level or a time-out has been reached. RXRDY remains cleared until the receiver FIFO is empty.

12.4 Functional Description

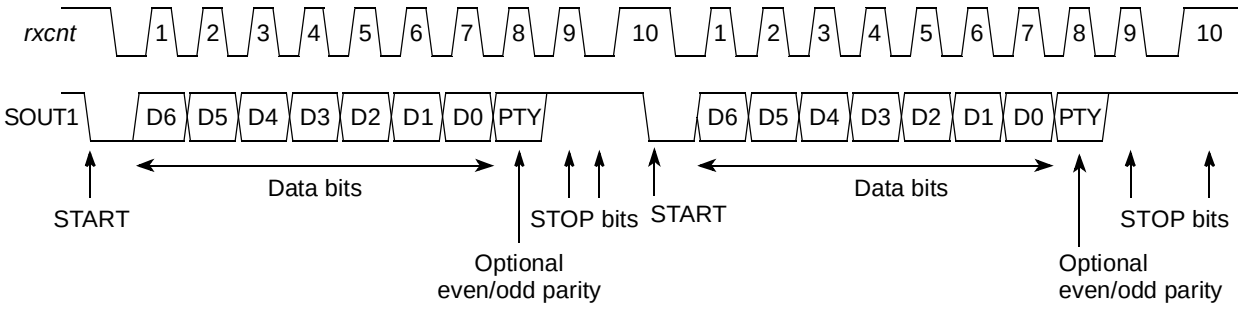
The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the CCB clock signal.

The transmitter accepts parallel data with a write access to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register, or into the transmitter FIFO—see Section 12.4.5, “FIFO Mode.” The transmitting registers convert the data to a serial bit stream, by inserting the appropriate START, STOP, and optional parity bits. Finally, the registers output a composite serial data stream on the channel transmitter serial data output (SOUT). The transmitter status may be polled or interrupt-driven.

The receiver accepts serial data on the channel receiver serial data input (SIN), converts the data into parallel format, and checks for START, STOP, and parity bits. In FIFO mode, the receiver removes the START, STOP, and parity bits and then transfers the assembled character from the receiver buffer, or receiver FIFO. This transfer occurs in response to a read of the UART receiver buffer register (URBR). The receiver status may be polled or interrupt driven.

12.4.1 Serial Interface

The UART bus is a serial, full-duplex, point-to-point bus as shown in Figure 12-16. Therefore, only two devices are attached to the same signals and there is no need for address or arbitration bus cycles.



Two 7-bit data transmissions with parity and 2-bit STOP transactions

Figure 12-16. UART Bus Interface Transaction Protocol Example

A standard UART bus transfer is composed of either three or four parts:

- START bit
- Data transfer bits (least-significant bit is first data bit on the bus)
- Parity bit (optional)
- STOP bits

An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to drive the bits on SOUT.

The following sections describe the four components of the serial interface, the baud-rate generator, local loop-back mode, different errors, and FIFO mode.

12.4.1.1 START Bit

A write to the transmitter holding register (UTHR) generates a START bit on the SOUT signal. Figure 12-16 shows that the START bit is defined as a logic 0. The START bit denotes the beginning of a new data transfer which is limited to the bit length programmed in the UART line control register (ULCR).When the bus is idle, SOUT is high.

12.4.1.2 Data Transfer

Each data transfer contains 5–8 bits of data. The ULCR data bit length for the transmitter and receiver UART devices must agree before a transfer begins; otherwise, a parity or framing error may occur. A transfer begins when UTHR is written. At that time a START bit is generated followed by 5–8 of the data bits previously written to the UTHR. The data bits are driven from the least significant to the most significant bits. After the parity and STOP bits, a new data transfer can begin if new data is written to the UTHR.

12.4.1.3 Parity Bit

The user has the option of using even, odd, no parity, or stick parity (see Section 12.3.1.7, “Line Control Registers (ULCR0, ULCR1).” Both the receiver and transmitter parity definition must agree before attempting to transfer data. When receiving data a parity error can occur if an unexpected parity value is detected. (See Section 12.3.1.9, “Line Status Registers (ULSR0, ULSR1).”)

12.4.1.4 STOP Bit

The transmitter device ends the write transfer by generating a STOP bit. The STOP bit is always high. The user can program the length of the STOP bit(s) in the ULCR. Both the receiver and transmitter STOP bit length must agree before attempting to transfer data. A framing error can occur if an invalid STOP bit is detected.

12.4.2 Baud-Rate Generator Logic

Each UART contains an independent programmable baud-rate generator, that is capable of taking the CCB clock input and dividing the input by any divisor from 1 to $2^{16} – 1$.

The baud rate is defined as the number of bits per second that can be sent over the UART bus. The formula for calculating baud rate is as follows:

$$\text{Baud rate} = (1/16) \times (\text{CCB clock frequency/divisor value})$$

Therefore, the output frequency of the baud-rate generator is 16 times the baud rate.

The divisor value is determined by the following two 8-bit registers to form a 16-bit binary number:

- UART divisor most significant byte register (UDMB)
- UART divisor least significant byte register (UDLB)

Upon loading either of the divisor latches, a 16-bit baud-rate counter is loaded.

The divisor latches must be loaded during initialization to ensure proper operation of the baud-rate generator. Both UART devices on the same bus must be programmed for the same baud-rate before starting a transfer.

The baud clock can be passed to the performance monitor by enabling the UAFR[BO] bit. This can be used to determine baud rate errors.

12.4.3 Local Loop-Back Mode

Local loop-back mode is provided for diagnostic testing. The data written to UTHR can be read from the receiver buffer register (URBR) of the same UART. In this mode, the MODEM control

register UMCR[RTS] is internally tied to the MODEM status register UMSR[CTS]. The transmitter SOUT is set to a logic 1 and the receiver SIN is disconnected. The output of the transmitter shift register is looped back into the receiver shift register input. The $\overline{\text{CTS}}$ (input signal) is disconnected, $\overline{\text{RTS}}$ is internally connected to $\overline{\text{CTS}}$, and the $\overline{\text{RTS}}$ (output signal) becomes inactive. In this diagnostic mode, data that is transmitted is immediately received. In local loop-back mode the transmit and receive data paths of the DUART can be verified. Note that in local loop-back mode, the transmit/receive interrupts are fully operational and can be controlled by the interrupt enable register (UIER).

12.4.4 Errors

The following sections describe framing, parity, and overrun errors which may occur while data is transferred on the UART bus. Each of the error bits are usually cleared, as described below, when the line status register (ULSR) is read.

12.4.4.1 Framing Error

When an invalid STOP bit is detected, a framing error occurs and ULSR[FE] is set. Note that only the first STOP bit is checked. In FIFO mode, ULSR[FE] is set when the character at the top of the FIFO detects a framing error. An attempt to re-synchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit. ULSR[FE] is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register.

12.4.4.2 Parity Error

A parity error occurs, and ULSR[PE] is set, when unexpected parity values are encountered while receiving data. In FIFO mode, ULSR[PE] is set when the character with the error is at the top of the FIFO. ULSR[PE] is cleared when ULSR is read or when a new character is loaded into the URBR.

12.4.4.3 Overrun Error

When a new (overwriting character) STOP bit is detected and the old character is lost, an overrun error occurs and ULSR[OE] is set. In FIFO mode, ULSR[OE] is set after the receiver FIFO is full (despite the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. Data in the FIFO is not overwritten; only the shift register data is overwritten. Therefore, the interrupt occurs immediately. ULSR[OE] is cleared when ULSR is read.

12.4.5 FIFO Mode

The UARTs use an alternate mode (FIFO mode) to relieve the processor core from excessive software overhead. The FIFO control register (UFCR) is used to enable and clear the receiver and transmitter FIFOs and set the FIFO receiver trigger level UFCR[RTL] to control the received data available interrupt UIER[ERDAI].

The UFCR also selects the type of DMA signaling. The UDSR[RXRDY] indicates the status of the receiver FIFO. The DMA status registers (UDSR[TXRDY]) indicate when the transmitter FIFO is full. When in FIFO mode, data written to UTHR is placed into the transmitter FIFO. The first byte written to UTHR is the first byte onto the UART bus.

12.4.5.1 FIFO Interrupts

In FIFO mode, the UIER[ERDAI] is set when a time-out interrupt occurs. When a receive data time-out occurs there is a maskable interrupt condition (through UIER[ERDAI]). See Section 12.3.1.4, “Interrupt Enable Register (UIER),” for more details on interrupt enables.

The interrupt ID register (UIIR) indicates if the FIFOs are enabled. Interrupt ID3 UIIR[IID3] bit is only set for FIFO mode interrupts. The character time-out interrupt occurs when no characters have been removed from or input to the receiver FIFO during the last four character times and there is at least one character in the receiver FIFO during this time. The character time-out interrupt (controlled by UIIR[IID]) is cleared when the URBR is read. See Section 12.3.1.5, “Interrupt ID Registers (UIIR0, UIIR1),” for more information.

The UIIR[FE] bits indicate if FIFO mode is enabled.

12.4.5.2 DMA Mode Select

The UDSR[RXRDY] bit reflects the status of the receiver FIFO or URBR. In mode 0 (UFCR[DMS] is cleared), UDSR[RXRDY] is cleared when there is at least one character in the receiver FIFO or URBR and it is set when there are no more characters in the receiver FIFO or URBR. This occurs regardless of the setting of the UFCR[FEN] bit. In mode 1 (UFCR[DMS] and UFCR[FEN] are set), UDSR[RXRDY] is cleared when the trigger level or a time-out has been reached and it is set when there are no more characters in the receiver FIFO.

The UDSR[TXRDY] bit reflects the status of the transmitter FIFO or UTHR. In mode 0 (UFCR[DMS] is cleared), UDSR[TXRDY] is cleared when there are no characters in the transmitter FIFO or UTHR and it is set after the first character is loaded into the transmitter FIFO or UTHR. This occurs regardless of the setting of the UFCR[FEN] bit. In mode 1 (UFCR[DMS] and UFCR[FEN] are set), UDSR[TXRDY] is cleared when there are no characters in the transmitter FIFO or UTHR and it is set when the transmitter FIFO is full.

See Section 12.3.1.13, “DMA Status Registers (UDSR0, UDSR1),” for a complete description of the USDR[RXRDY] and USDR[TXRDY] bits.

12.4.5.3 Interrupt Control Logic

An interrupt is active when DUART interrupt ID register bit 0 (UIIR[0]), is cleared. The interrupt enable register (UIER) is used to mask specific interrupt types. For more details refer to the description of UIER in Section 12.3.1.4, "Interrupt Enable Register (UIER)."

When the interrupts are disabled in UIER, polling software can not use UIIR[0] to determine whether the UART is ready for service. The software must monitor the appropriate bits in the line status (ULSR) and/or the MODEM status (UMSR) registers. UIIR[0] can be used for polling if the interrupts are enabled in UIER.

12.5 DUART Initialization/Application Information

The following requirements must be met for DUART accesses:

- All DUART registers must be mapped to a cache-inhibited area.
- All DUART registers are 1 byte wide. Reads and writes to these registers must be byte-wide operations.

A system reset puts the DUART registers to a default state. Before the interface can transfer serial data, the following initialization steps are recommended:

1. Update the programmable interrupt controller (PIC) DUART channel interrupt vector source registers.
2. Set data attributes and control bits in the ULCR, UFCR, UAFR, UMCR, UDLB, and UDMB.
3. Set the data attributes and control bits of the external MODEM or peripheral device.
4. Set the interrupt enable register (UIER).
5. To start a write transfer, write to the UTHR.
6. Poll UIIR if the interrupts generated by the DUART are masked.

Chapter 13
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.

13.1 Introduction

Figure 13-1 is a functional block diagram of the LBC, which supports three interfaces: GPCM, UPM, and SDRAM controller.

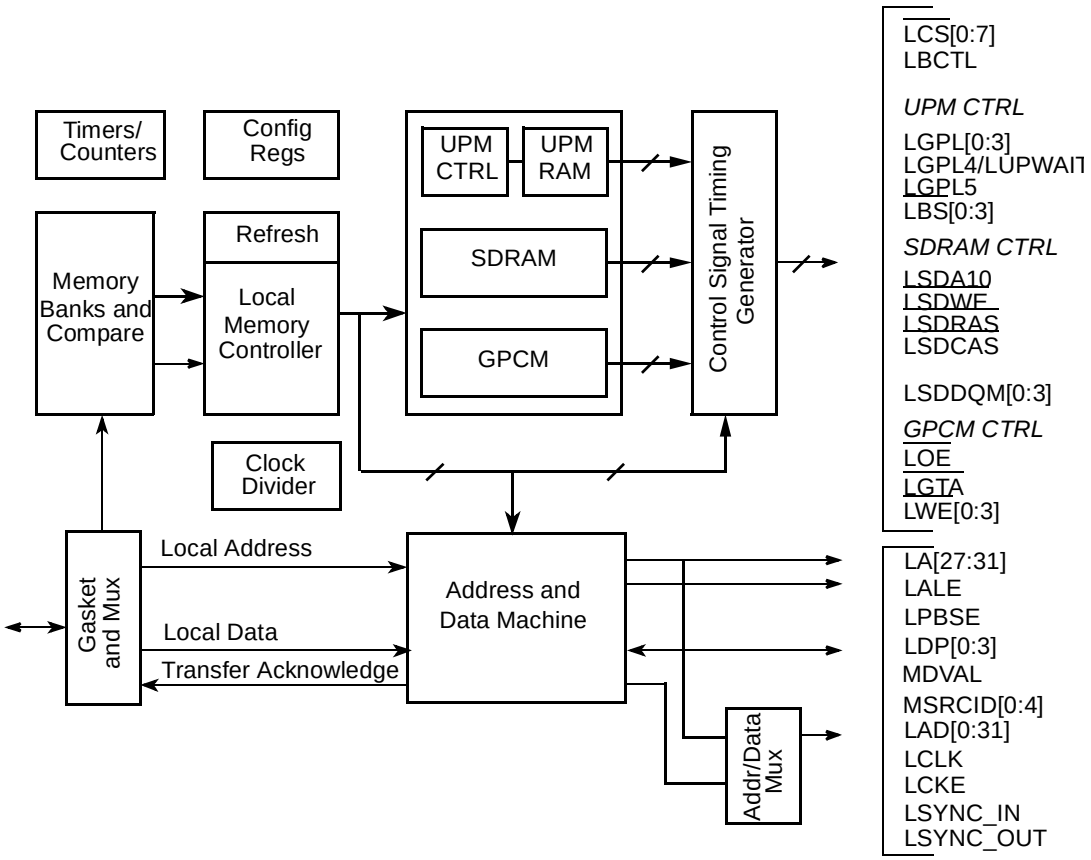


Figure 13-1. Local Bus Controller Block Diagram

13.1.1 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.

13.1.2 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

13.1.3 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.

13.1.3.1 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.

13.1.3.2 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 20.4.4, "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].

13.1.4 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.

13.1.5 References

- *MPC8260 PowerQUICC II Family Reference Manual*, Chapters 4, 6, and 10 (order no. MPC8260UM).

13.2 External Signal Descriptions

Table 13-1 contains a list of external signals related to the LBC and summarizes their function.

Table 13-1. 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 13-2 shows the detailed external signal descriptions for the LBC.

Table 13-2. Local Bus Controller Detailed Signal Descriptions

Signal	I/O	Description	
LALE	O	External address latch enable. The local bus memory controller provides control for an external address latch, which allows address and data to be multiplexed on the device pins.	
		State Meaning	Asserted/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 13.4.4.4, "RAM Array," for programming details about $\overline{\text{LBS}}$[0:3].
		Timing	Assertion/Negation—See Section 13.4.2, "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 13-2. 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.

Table 13-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
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.
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 13-19 on page 13-32), 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.

Table 13-2. Local Bus Controller Detailed Signal Descriptions (continued)

Signal	I/O	Description	
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

13.3 Memory Map/Register Definition

Table 13-3 shows the memory mapped registers of the LBC. Undefined 4-byte address spaces within offset 0x000–0xFFF are reserved.

Table 13-3. Local Bus Controller Memory Map

Address Offset	Use	Access	Reset	Section/Page
0x0_5000	BR0—Base register 0	R/W	0x0000_*nn*01[1]	13.3.1.1/13-11
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			

Table 13-3. Local Bus Controller Memory Map (continued)

Address Offset	Use	Access	Reset	Section/Page
0x0_5004	OR0—Options register 0	R/W	0x0000_0FF7	13.3.1.2/13-12
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	13.3.1.3/13-18
0x0_5070	MAMR—UPMA mode register	R/W	0x0000_0000	13.3.1.4/13-19
0x0_5074	MBMR—UPMB mode register	R/W	0x0000_0000	13.3.1.4/13-19
0x0_5078	MCMR—UPMC mode register	R/W	0x0000_0000	13.3.1.4/13-19
0x0_5084	MRTPR—Memory refresh timer prescaler register	R/W	0x0000_0000	13.3.1.5/13-21
0x0_5088	MDR—UPM data register	R/W	0x0000_0000	13.3.1.6/13-22
0x0_5094	LSDMR—SDRAM mode register	R/W	0x0000_0000	13.3.1.7/13-22
0x0_50A0	LURT—UPM refresh timer	R/W	0x0000_0000	13.3.1.8/13-24
0x0_50A4	LSRT—SDRAM refresh timer	R/W	0x0000_0000	13.3.1.9/13-25
0x0_50B0	LTESR—Transfer error status register	Read/Bit reset	0x0000_0000	13.3.1.10/13-26
0x0_50B4	LTEDR—Transfer error disable register	R/W	0x0000_0000	13.3.1.11/13-27
0x0_50B8	LTEIR—Transfer error interrupt register	R/W	0x0000_0000	13.3.1.12/13-28
0x0_50BC	LTEATR—Transfer error attributes register	R/W	0x0000_0000	13.3.1.13/13-29
0x0_50C0	LTEAR—Transfer error address register	R/W	0x0000_0000	13.3.1.14/13-30
0x0_50D0	LBCR—Configuration register	R/W	0x0000_0000	13.3.1.15/13-31
0x0_50D4	LCRR—Clock ratio register	R/W	0x8000_0008	13.3.1.16/13-32

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

13.3.1 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 13-3 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.

13.3.1.1 Base Registers (BR0–BR7)

The base registers (BR*n*), shown in Figure 13-2, 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 13-2. Base Registers (BR*n*)

Table 13-4 describes BR*n* fields.

Table 13-4. 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
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

Table 13-4. BR*n* Field Descriptions (continued)

Bits	Name	Description
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.

13.3.1.2 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.

13.3.1.2.1 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 13-5 shows memory bank sizes from 256 Kbytes to 4 Gbytes.

Table 13-5. 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

13.3.1.2.2 Option Registers (OR*n*)—GPCM Mode

Figure 13-3 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 13-3. Option Registers (OR*n*) in GPCM Mode

Table 13-6 describes OR*n* fields for GPCM mode.

Table 13-6. 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 13-6. 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 13-23 and Table 13-24 for LCRR[CLKDIV] = 4 or 8, Table 13-25 and Table 13-26 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 13-6. OR*n*—GPCM Field Descriptions (continued)

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

13.3.1.2.3 Option Registers (OR*n*)—UPM Mode

Figure 13-4 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects a UPM machine.

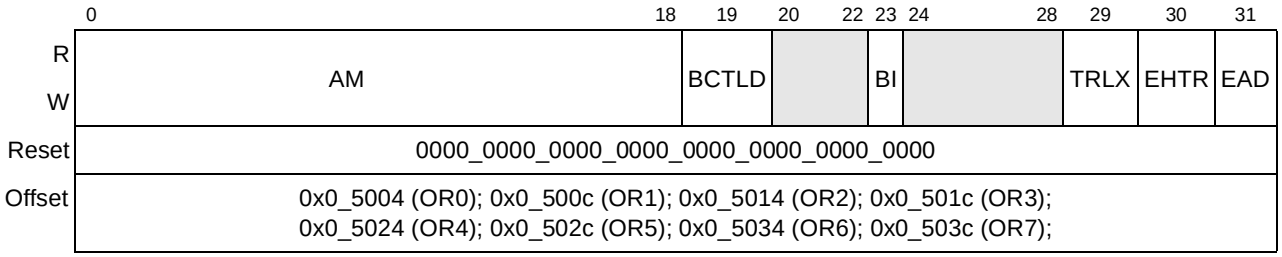


Figure 13-4. Option Registers (OR*n*) in UPM Mode

Table 13-7 describes BR*n* fields for UPM mode.

Table 13-7. 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 13-7. 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. **TRLX / EHTR / Meaning** 0 / 0 — The memory controller generates normal timing. No additional cycles are inserted. 0 / 1 — 1 idle clock cycle is inserted. 1 / 0 — 4 idle clock cycles are inserted. 1 / 1 — 8 idle clock cycles are inserted.
31	EAD	External address latch delay. Allow extra bus clock cycles when using external address latch (LALE). 0 No additional bus clock cycles (LALE asserted for one bus clock cycle only) 1 Extra bus clock cycles are added (LALE is asserted for the number of bus clock cycles specified by LCRR[EADC]).

13.3.1.2.4 Option Registers (OR*n*)—SDRAM Mode

Figure 13-5 shows the bit fields for OR*n* when the corresponding BR*n*[MSEL] selects the SDRAM machine.

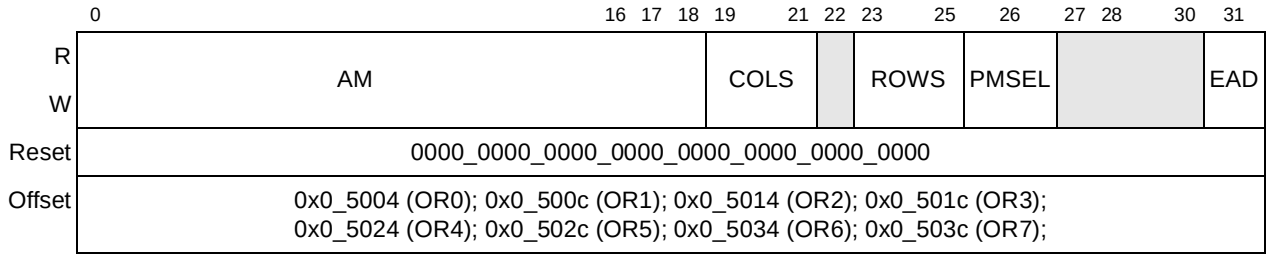


Figure 13-5. Option Registers (OR*n*) in SDRAM Mode

Table 13-8 describes BR*n* fields for SDRAM mode.

Table 13-8. 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]).

13.3.1.3 UPM Memory Address Register (MAR)

Figure 13-6 shows the fields of the UPM memory address register (MAR).

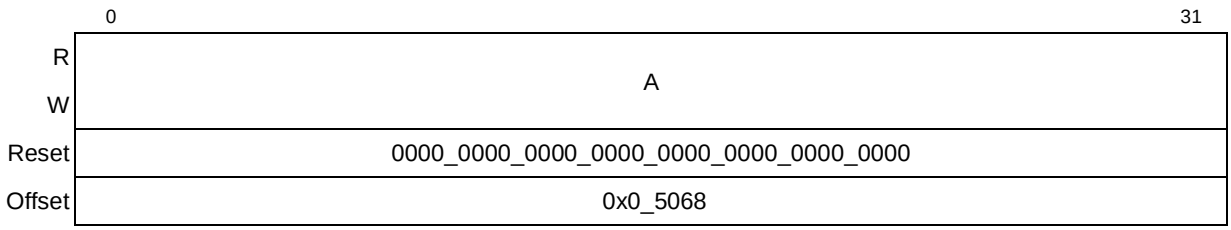


Figure 13-6. UPM Memory Address Register (MAR)

Table 13-9 describes the MAR field.

Table 13-9. 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.

13.3.1.4 UPM Mode Registers (M*x*MR)

The UPM machine mode registers (MAMR, MBMR and MCMR), shown in Figure 13-7, contain the configuration for the three UPMs.

	0	1	2	3	4	5	7	8	9	10	12	13	14	17	18	21	22	25	26	31
R W		RFEN	OP		UWPL	AM		DS		G0CL		GPL4	RLF		WLF		TLF		MAD	
Reset	0000_0000_0000_0000_0000_0000_0000_0000																			
Offset	0x0_5070 (MAMR); 0x0_5074 (MBMR); 0x0_5078 (MCMR)																			

Figure 13-7. UPM Mode Registers (M*x*MR)

Table 13-10 describes UPM mode fields.

Table 13-10. 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 13.4.4.2, “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 13-10. 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 13-30 on page 13-69.
<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 13-10. 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*.

13.3.1.5 Memory Refresh Timer Prescaler Register (MRTPR)

The refresh timer prescaler register (MRTPR), shown in Figure 13-8, is used to divide the system clock to provide the SDRAM and UPM refresh timers clock.

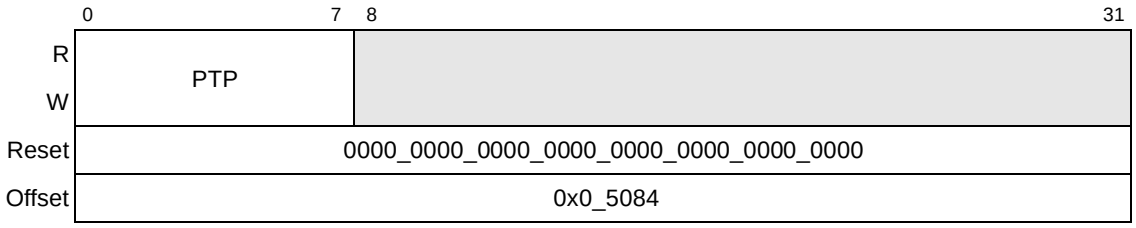


Figure 13-8. Memory Refresh Timer Prescaler Register (MRTPR)

Table 13-11 describes MRTPR fields.

Table 13-11. MRTPR Field Descriptions

Bits	Name	Description
0–7	PTP	Refresh timers prescaler. Determines the period of the refresh timers input clock. The system clock is divided by PTP except when the value is 0000_0000, which represents the maximum divider of 256.
8–31	—	Reserved

13.3.1.6 UPM Data Register (MDR)

The memory data register (MDR), shown in Figure 13-9, 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.

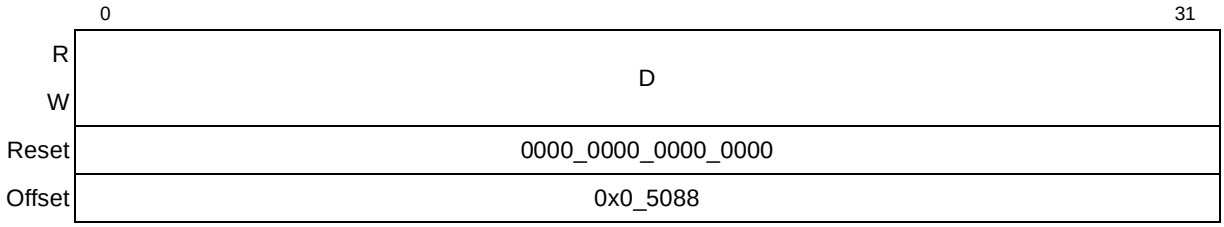


Figure 13-9. UPM Data Register (MDR)

Table 13-12 describes MDR[D].

Table 13-12. 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).

13.3.1.7 SDRAM Machine Mode Register (LSDMR)

The local bus SDRAM mode register (LSDMR), shown in Figure 13-10, is used to configure operations pertaining to SDRAM.

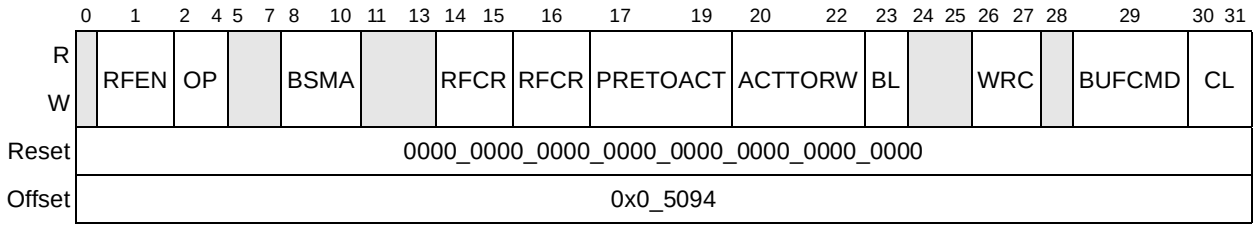


Figure 13-10. SDRAM Machine Mode Register (LSDMR)

Table 13-12 describes LSDMR fields.

Table 13-13. LSDMR Field Descriptions

<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 13-13. LSDMR Field Descriptions (continued)

Bits	Name	Description
20–22	ACTTORW	Defines the earliest timing for READ/WRITE command after an ACTIVATE command (number of bus clock cycle wait states). 000 8 100 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 LCS*n*, LCKE, LALE and 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 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 13-22 on page 13-32. 01 1 10 2 11 3

13.3.1.8 UPM Refresh Timer (LURT)

The UPM refresh timer (LURT), shown in Figure 13-11, 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.

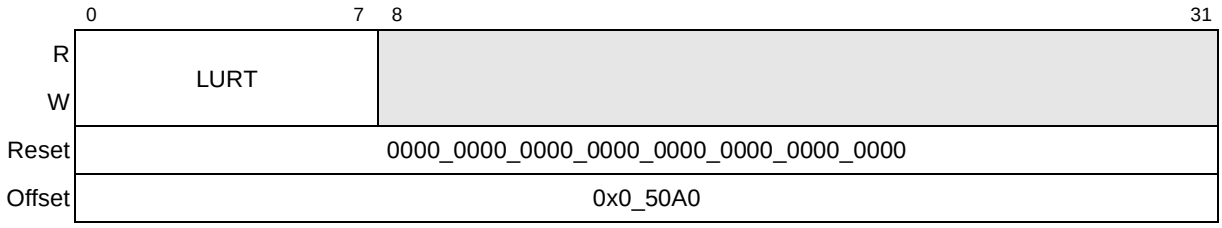


Figure 13-11. UPM Refresh Timer (LURT)

Table 13-14 describes LURT fields.

Table 13-14. 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

13.3.1.9 SDRAM Refresh Timer (LSRT)

The SDRAM refresh timer (LSRT), shown in Figure 13-12, 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.

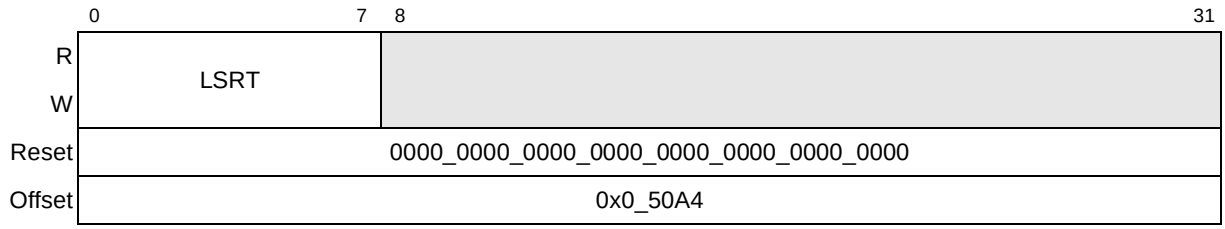


Figure 13-12. LSRT SDRAM Refresh Timer (LSRT)

Table 13-15 describes LSRT fields.

Table 13-15. LSRT Field Descriptions

Bits	Name	Description
0–7	LSRT	SDRAM refresh timer period. Determines, along with the timer prescaler (MRTPR), the timer period according to the following equation: $TimerPeriod = \frac{LSRT}{\left(\frac{Fsystemclock}{MRTPR[PTP]}\right)}$ **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

13.3.1.10 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 13-13, 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.

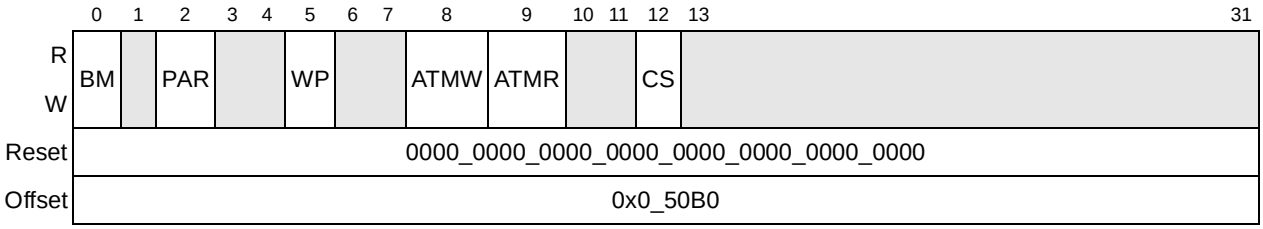


Figure 13-13. Transfer Error Status Register (LTESR)

Table 13-16 describes LTESR fields.

Table 13-16. LTESR Field Descriptions

Bits	Name	Description
0	BM	Bus monitor time-out 0 No local bus monitor time-out occurred. 1 Local bus monitor time-out occurred. No data beat was acknowledged on the bus within LBCR[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 13-16. 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

13.3.1.11 Transfer Error Check Disable Register (LTEDR)

The transfer error check disable register (LTEDR), shown in Figure 13-14, 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.

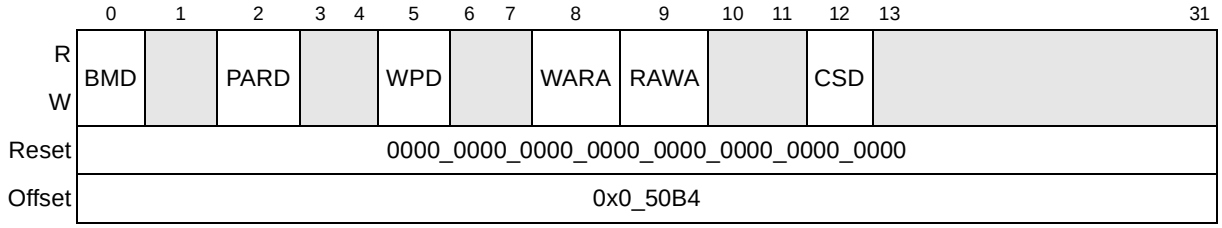


Figure 13-14. Transfer Error Check Disable Register (LTEDR)

Table 13-17 describes LTEDR fields.

Table 13-17. LTEDR Field Descriptions

Bits	Name	Description
0	BMD	Bus monitor disable 0 Bus monitor is enabled 1 Bus monitor is disabled
1	—	Reserved

Table 13-17. 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

13.3.1.12 Transfer Error Interrupt Enable Register (LTEIR)

The transfer error interrupt enable register (LTEIR), shown in Figure 13-15, 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.

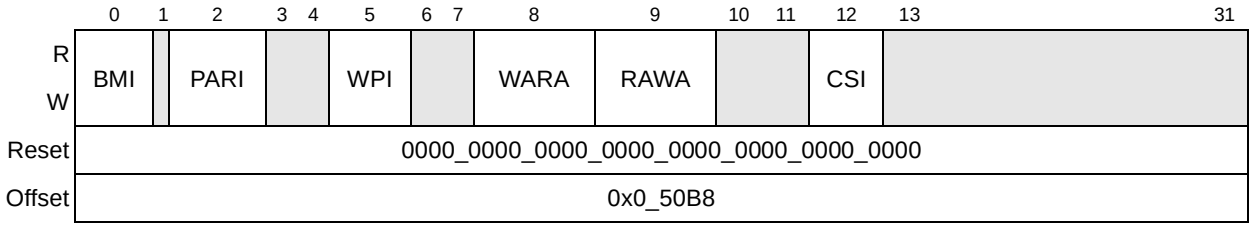


Figure 13-15. Transfer Error Interrupt Enable Register (LTEIR)

Table 13-18 describes LTEIR fields.

Table 13-18. LTEIR Field Descriptions

Bits	Name	Description
0	BMI	Bus monitor error interrupt enable 0 Bus monitor error reporting is disabled. 1 Bus monitor error reporting is enabled.
1	—	Reserved
2	PARI	Parity error interrupt enable. 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

13.3.1.13 Transfer Error Attributes Register (LTEATR)

Figure 13-16 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.

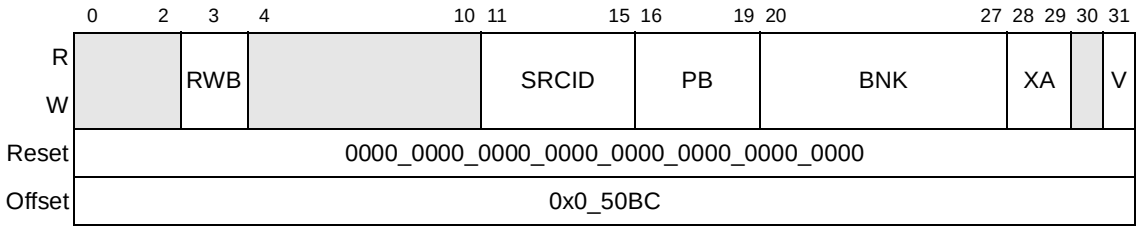


Figure 13-16. Transfer Error Attributes Register (LTEATR)

Table 13-19 describes LTEATR fields.

Table 13-19. LTEATR Field Descriptions

Bits	Name	Description
0–2	—	Reserved
3	RWB	Transaction type for the error 0 The transaction for the error was a write transaction. 1 The transaction for the error was a read transaction.
4–10	—	Reserved
11–15	SRCID	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

13.3.1.14 Transfer Error Address Register (LTEAR)

The transfer error address register (LTEAR) is shown in Figure 13-17.

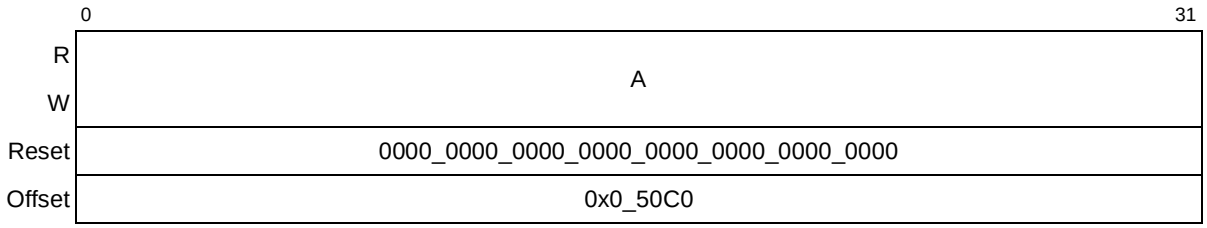


Figure 13-17. Transfer Error Address Register (LTEAR)

Table 13-20 describes LTEAR[A].

Table 13-20. 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.

13.3.1.15 Local Bus Configuration Register (LBCR)

The local bus configuration register (LBCR) is shown in Figure 13-18.

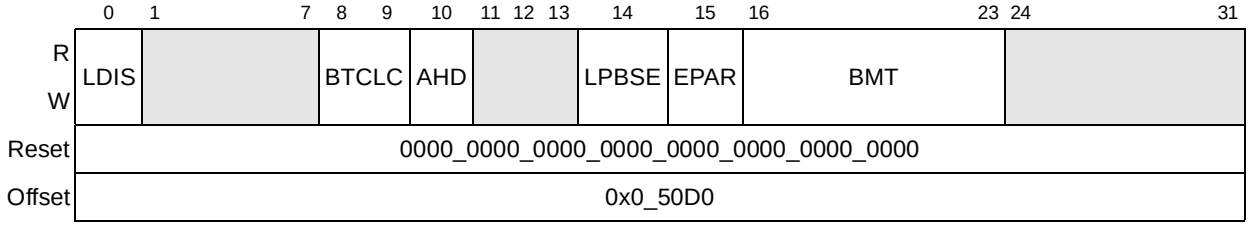


Figure 13-18. Local Bus Configuration Register

Table 13-21 describes LBCR fields.

Table 13-21. LBCR Field Descriptions

Bits	Name	Description
0	LDIS	Local bus disable 0 Local bus is enabled 1 Local bus is disabled. No internal transactions will be acknowledged.
1–7	—	Reserved
8–9	BCTLC	Defines the use of LBCTL 00 LBCTL is used as W/R control for GPCM or UPM accesses (buffer control). 01 LBCTL is used as LOE for GPCM accesses only. 10 LBCTL is used as 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 LGTA/LGPL4/LUPWAIT/LPBSE pin. 0 Parity byte select is disabled. LGTA/LGPL4/LUPWAIT/LPBSE pin is available for memory control as LGPL4 (output) or LGTA/LUPWAIT (input). 1 Parity byte select is enabled. LGTA/LGPL4/LUPWAIT/LPBSE pin is dedicated as the parity byte select output, and 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 13-21. 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

13.3.1.16 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.

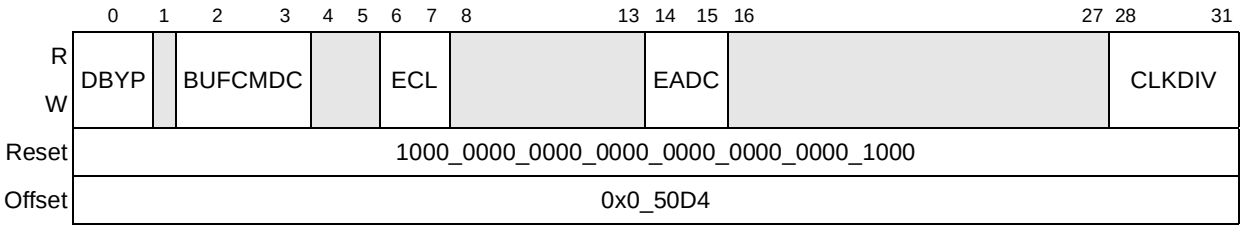


Figure 13-19. Clock Ratio Register (LCRR)

Table 13-22 describes LCRR fields.

Table 13-22. LCRR Field Descriptions

Bits	Name	Description
0	DBYP	DLL bypass. This bit should be set when using low bus clock frequencies if the DLL is unable to lock. When in DLL bypass mode, incoming data is captured in the middle of the bus clock cycle. 0 The DLL is enabled. 1 The DLL is bypassed.
1	—	Reserved
2–3	BUFCMDC	Additional delay cycles for SDRAM control signals. Defines the number of cycles to be added for each SDRAM command when LSDMR[BUFCMD] = 1. 00 4 01 1 10 2 11 3
4–5	—	Reserved
6–7	ECL	Extended CAS latency. Determines the extended CAS latency for SDRAM accesses when LSDMR[CL] = 00. 00 4 01 5 10 6 11 7
8–13	—	Reserved

Table 13-22. 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. 0000–0001Reserved 0010 2 0011 Reserved 0100 4 0101–0111Reserved 1000 8 1001–1111Reserved

13.4 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.

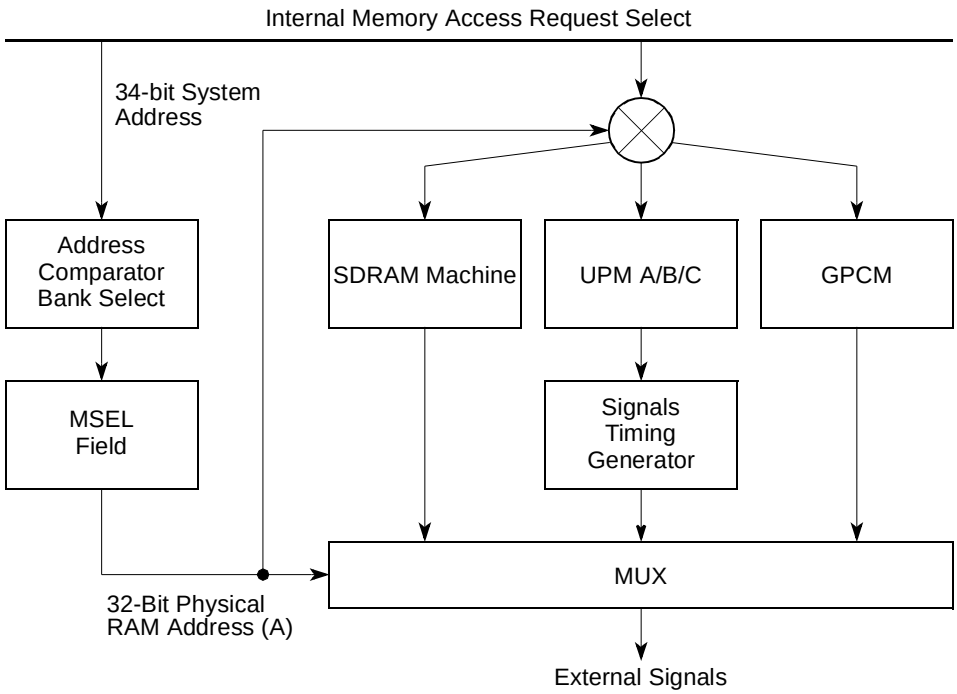


Figure 13-20. Basic Operation of Memory Controllers in the LBC

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 13-20. If a bank match occurs, the corresponding machine (GPCM, SDRAM or UPM) then takes ownership of the external signals that control the access and maintains control until the transaction ends.

13.4.1 Basic Architecture

The following sections describe the basic architecture of the LBC.

13.4.1.1 Address and Address Space Checking

The defined base addresses are written to the BR*n* registers, while the corresponding address masks are written to the OR*n* registers. Each time a local bus access is requested, the internal transaction address is compared with each bank. Addresses are decoded by comparing the 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).

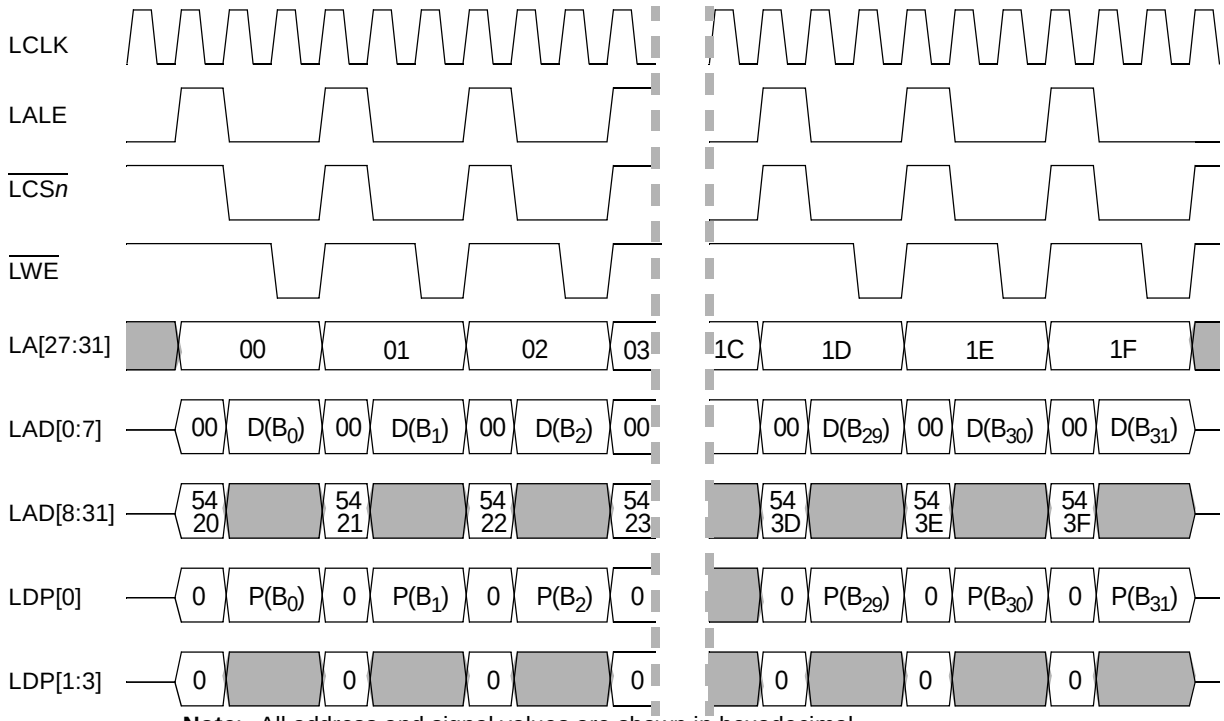
13.4.1.2 External Address Latch Enable Signal (LALE)

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 13-21 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 13-21. Example of 8-Bit GPCM Writing 32 Bytes to Address 0x5420

13.4.1.3 Data Transfer Acknowledge (TA)

The three memory controllers in the LBC generate an internal transfer acknowledge signal, TA, to allow data on LAD[0:31] to be either sampled (for reads) or changed (on writes). The data sampling/data change always occurs at the end of the bus cycle in which 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 13-22 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.

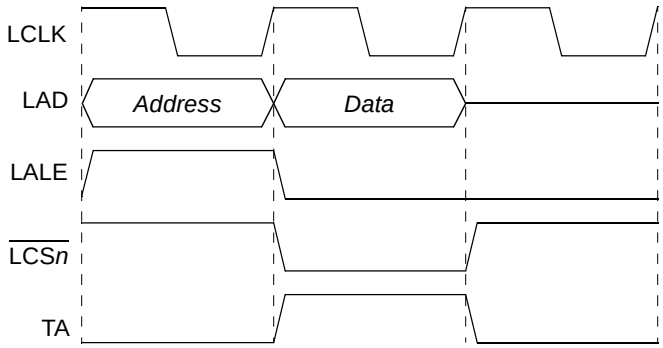


Figure 13-22. Basic LBC Bus Cycle with LALE, TA, and $\overline{\text{LCS}n}$

13.4.1.4 Data Buffer Control (LBCTL)

The memory controller provides a data buffer control signal for the local bus (LBCTL). This signal is activated when a GPCM or UPM controlled bank is accessed. 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.

13.4.1.5 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).

13.4.1.6 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).

13.4.1.7 Bus Monitor

A bus monitor is provided to ensure that each bus cycle is terminated within a reasonable (user defined) period. When a transaction starts, the bus monitor starts counting down from the time-out value (LBCR[BMT]) until a data beat is acknowledged on the bus. It then reloads the time-out value and resumes the 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.

13.4.2 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 13-23 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.

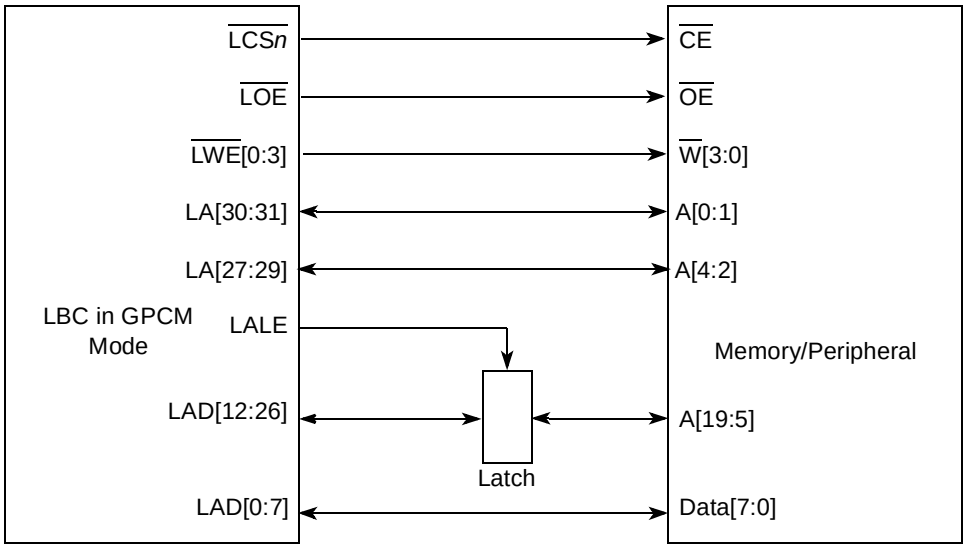


Figure 13-23. Local Bus to GPCM Device Interface

Figure 13-24 shows $\overline{\text{LCS}}$ as defined by the setup time required between the address lines and $\overline{\text{CE}}$. The user can configure OR*n*[ACS] to specify $\overline{\text{LCS}}$ to meet this requirement.

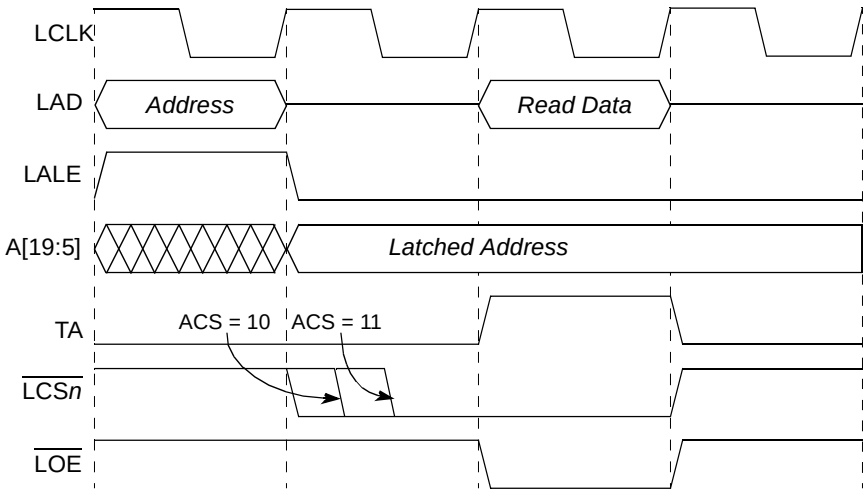


Figure 13-24. GPCM Basic Read Timing (XACS = 0, ACS = 1x, TRLX = 0, CLKDIV = 4,8)

13.4.2.1 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 13-23

shows signal behavior and system response for a write access with LCRR[CLKDIV] = 4 or LCRR[CLKDIV] = 8. Table 13-24 shows the signal behavior and system response for a read access with LCRR[CLKDIV] = 4 or LCRR[CLKDIV] = 8. Table 13-25 and Table 13-26 show the write and read signal behavior, respectively, when LCRR[CLKDIV] = 2.

Table 13-23. GPCM Write Control Signal Timing for LCRR[CLKDIV] = 4 or 8

Option Register Attributes				Signal Behavior (Bus Clock Cycles)			
TRLX	XACS	ACS	CSNT	Address to $\overline{\text{LCS}n}$ 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	1/4	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 13-24. GPCM Read Control Signal Timing for LCRR[CLKDIV] = 4 or 8

Option Register Attributes				Signal Behavior (bus clock cycles)		
TRLX	EHTR	XACS	ACS	Address to $\overline{\text{LCS}n}$ 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 13-25. GPCM Write Control Signal Timing for LCRR[CLKDIV] = 2

Option Register Attributes				Signal Behavior (bus clock cycles)			
TRLX	XACS	ACS	CSNT	Address to $\overline{\text{LCS}n}$ 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 13-26. GPCM Read Control Signal Timing for LCRR[CLKDIV] = 2

Option Register Attributes				Signal Behavior (Bus Clock cycles)		
TRLX	EHTR	XACS	ACS	Address to $\overline{\text{LCS}n}$ 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.

13.4.2.2 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 13-24 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.

13.4.2.2.1 Programmable Wait State Configuration

The GPCM supports internal generation of transfer acknowledge. It allows between zero and 30 wait states to be added to an access by programming 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.

13.4.2.2.2 Chip-Select and Write Enable Negation Timing

Figure 13-23 shows a basic connection between the local bus and a static memory device. In this case, $\overline{\text{LCS}n}$ 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.

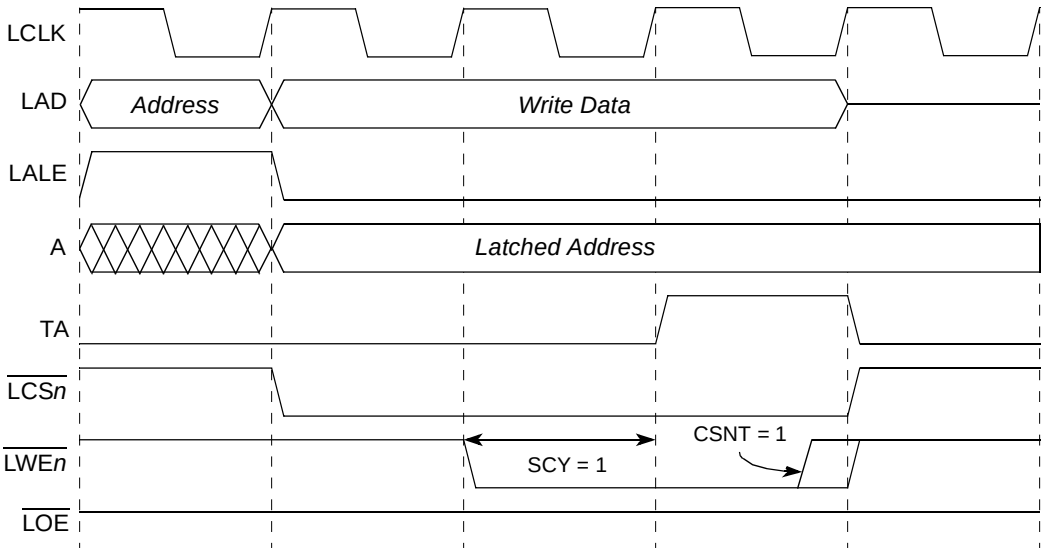


Figure 13-25. GPCM Basic Write Timing (XACS = 0, ACS = 00, CSNT = 1, SCY = 1, TRLX = 0, CLKDIV = 4 or 8)

As Figure 13-25 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 13-25). 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 13-25. If LCRR[CLKDIV] = 2, $\overline{\text{LWE}n}$ is negated either coincident with $\overline{\text{LCS}n}$ or one cycle earlier.

13.4.2.2.3 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.

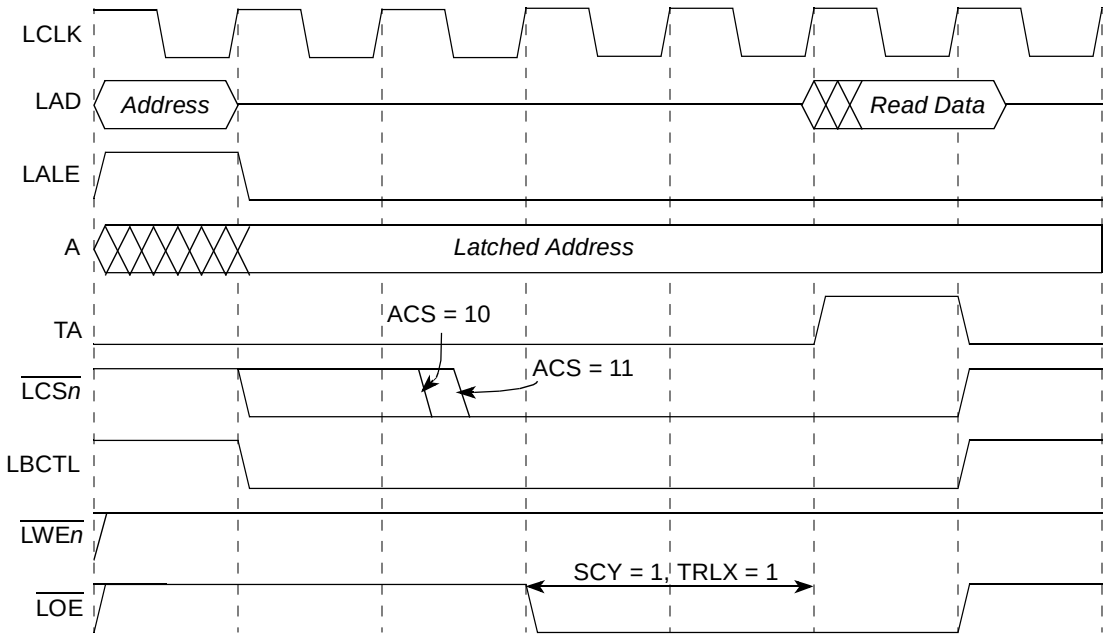


Figure 13-26. GPCM Relaxed Timing Read (XACS = 0, ACS = 1x, SCY = 1, CSNT = 0, TRLX = 1, CLKDIV = 4 or 8)

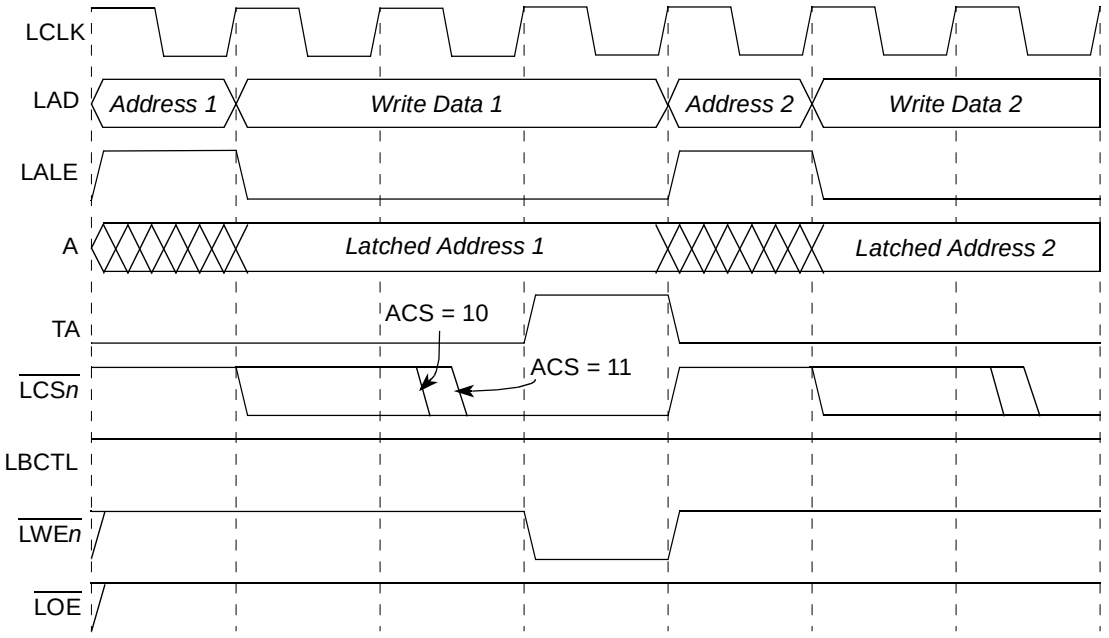


Figure 13-27. GPCM Relaxed Timing Back-to-Back Writes (XACS = 0, ACS = 1x, SCY = 0, CSNT = 0, TRLX = 1, CLKDIV = 4 or 8)

Figure 13-26 and Figure 13-27 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 13-27 also shows address and data multiplexing on LAD[0:31] for a pair of writes issued consecutively.

When TRLX and CSNT are set in a write access, the $\overline{\text{LWE}}$[0:3] strobe signals are negated one clock earlier than in the normal case, as shown in Figure 12-28 and Figure 12-29. If ACS ≠ 00, $\overline{\text{LCS}n}$ is also negated one clock earlier.

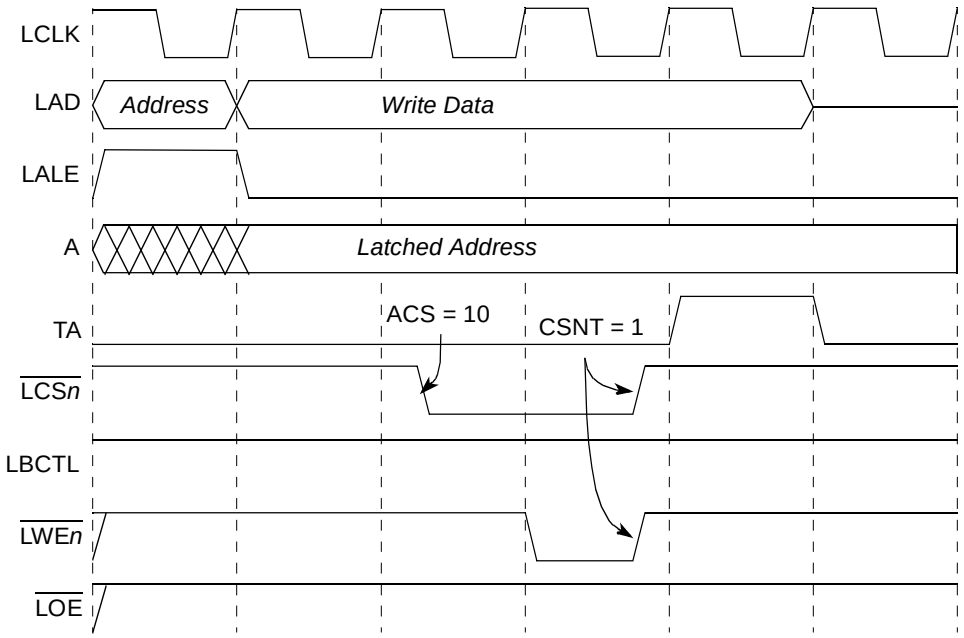


Figure 13-28. GPCM Relaxed Timing Write (XACS = 0, ACS = 10, SCY = 0, CSNT = 1, TRLX = 1, CLKDIV = 4 or 8)

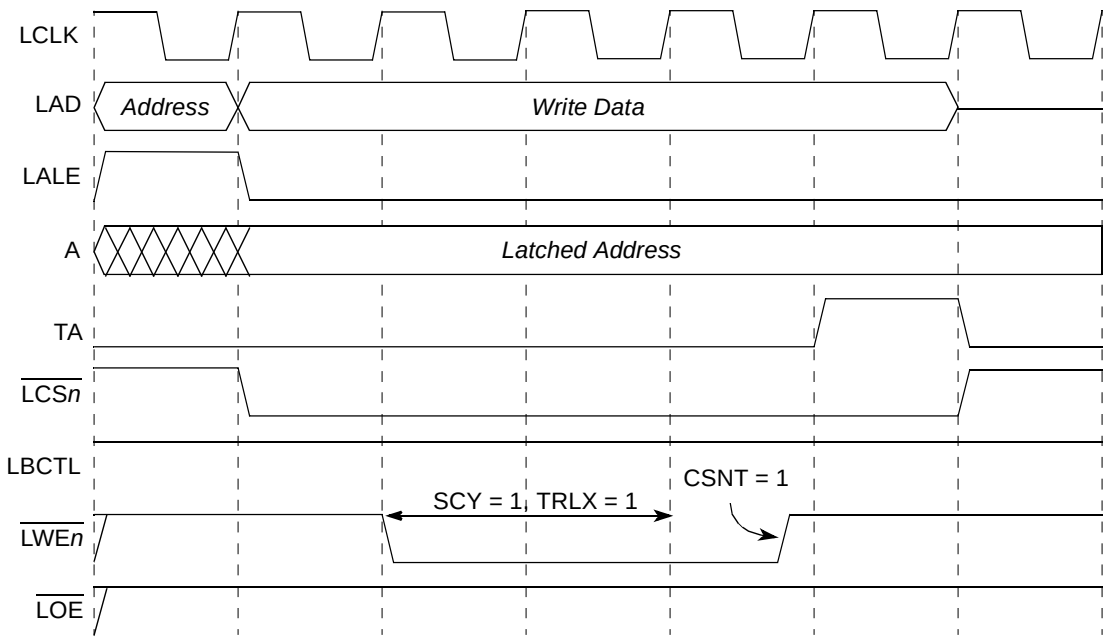


Figure 13-29. GPCM Relaxed Timing Write (XACS = 0, ACS = 00, SCY = 1, CSNT = 1, TRLX = 1, CLKDIV = 4 or 8)

13.4.2.2.4 Output Enable ($\overline{\text{LOE}}$) Timing

The timing of the $\overline{\text{LOE}}$ is affected only by TRLX. It always asserts and negates on the rising edge of the bus clock. $\overline{\text{LOE}}$ asserts either on the rising edge of the bus clock after $\overline{\text{LCS}n}$ 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

13.4.2.2.5 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to disable their data bus drivers on read accesses should choose some combination of 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 13-6 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].

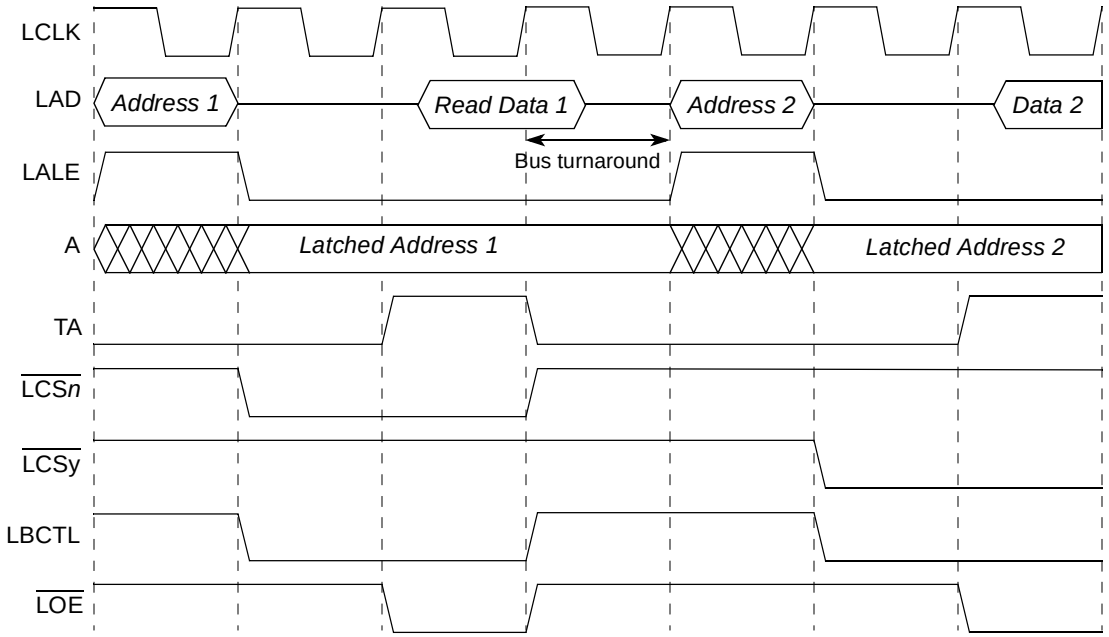


Figure 13-30. GPCM Read Followed by Read (TRLX = 0, EHTR = 0, Fastest Timing)

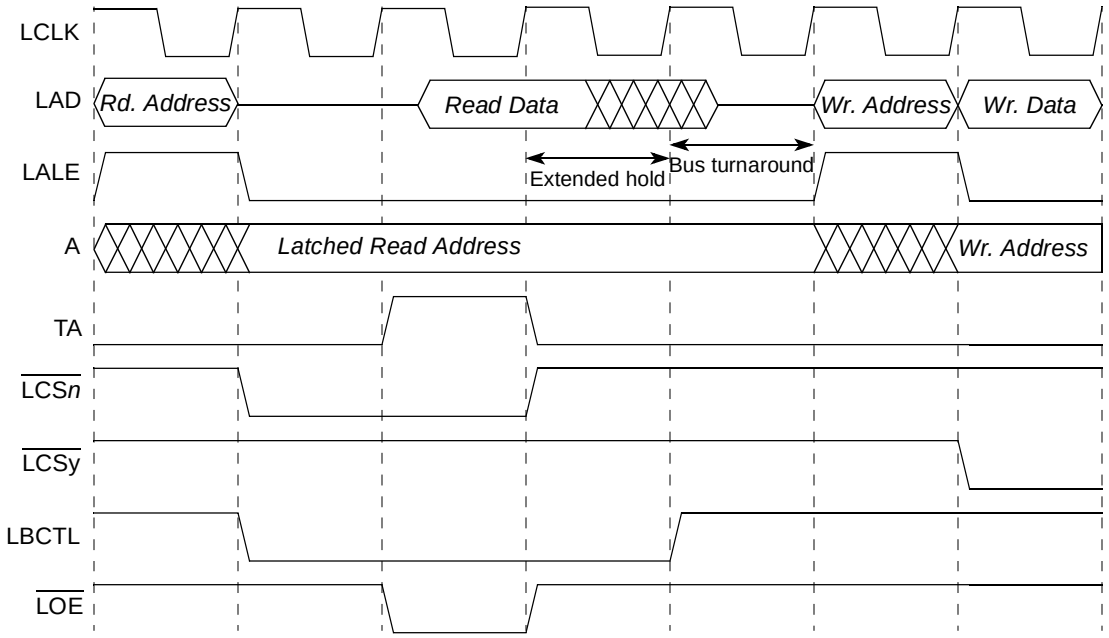


Figure 13-31. GPCM Read Followed by Write (TRLX = 0, EHTR = 1, 1-Cycle Extended Hold Time on Reads)

13.4.2.3 External Access Termination ($\overline{\text{LGTA}}$)

External access termination is supported by the GPCM using the asynchronous $\overline{\text{LGTA}}$ input signal, which is synchronized and sampled internally by the local bus. If, during assertion of $\overline{\text{LCS}n}$,

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 13-32.

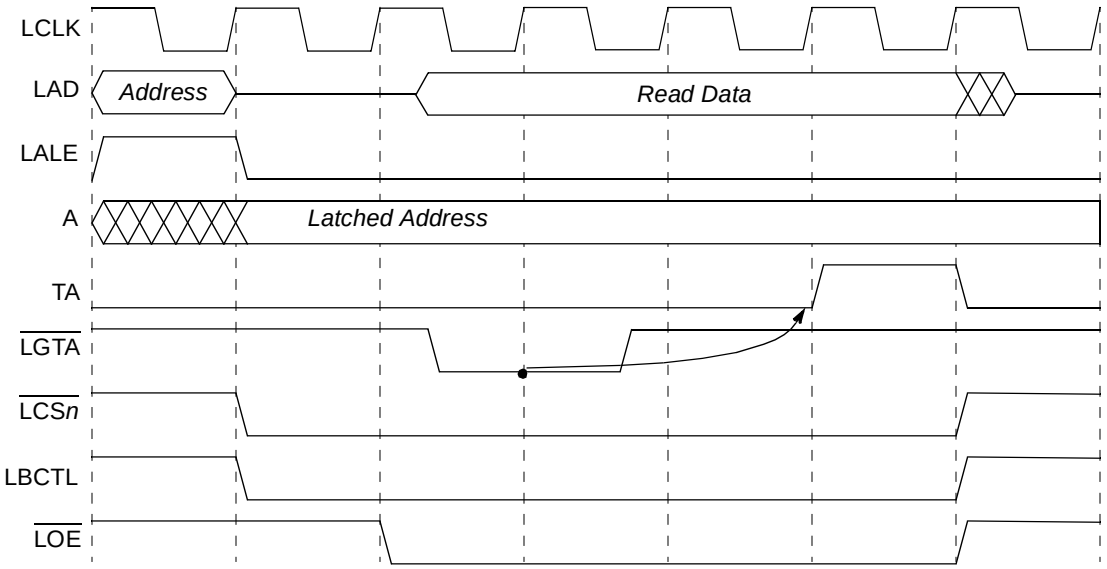


Figure 13-32. External Termination of GPCM Access

13.4.2.4 Boot Chip-Select Operation

Boot chip-select operation allows address decoding for a boot ROM before system initialization. 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 13-27 describes the initial values of the boot bank in the memory controller.

Table 13-27. Boot Bank Field Values after Reset

Register	Field	Setting
BR0	BA 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

13.4.3 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.

13.4.3.1 Supported SDRAM Configurations

The memory controller supports any SDRAM configuration with the restrictions that all SDRAM devices that reside on the bus should have the same port size and timing parameters (as defined in LSDMR). Figure 13-33 shows an example connection between the LBC and a 32-bit SDRAM device with 12 address lines. Note that address signals A[4:0] of the SDRAM connect directly to LA[27:31], address pin A10 connects to the LBCs dedicated LSDA10 signal, while the remaining address bits (except A10) are latched from LAD[20:26].

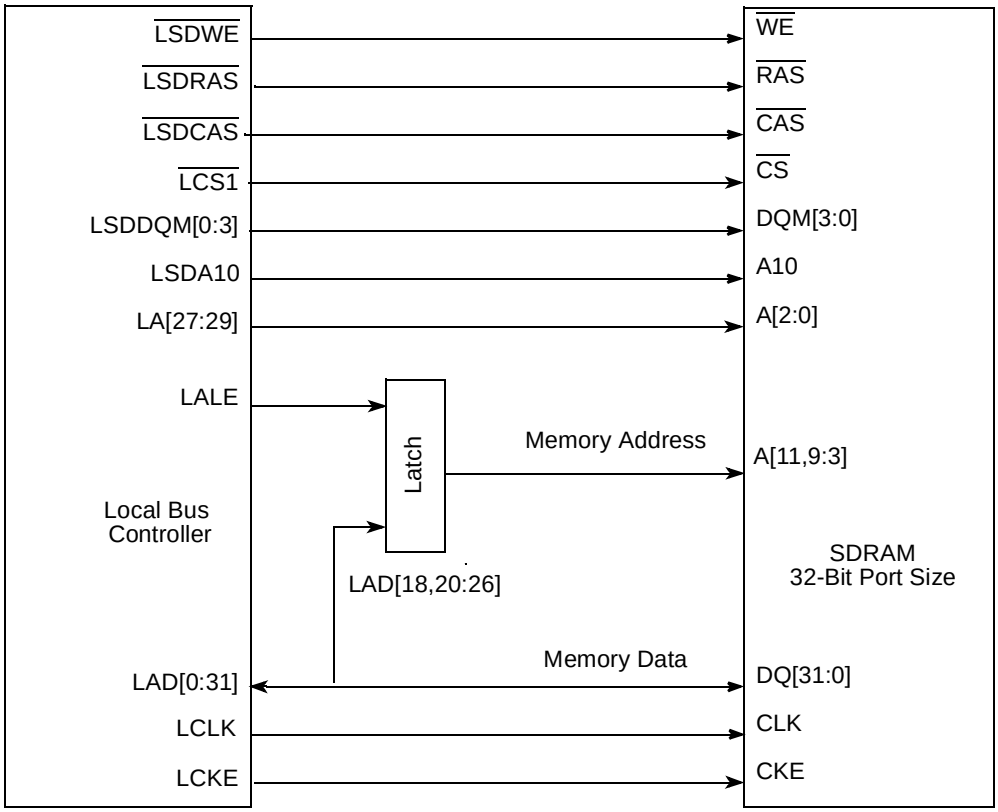


Figure 13-33. Connection to a 32-Bit SDRAM with 12 Address Lines

13.4.3.2 SDRAM Power-On Initialization

Following a system reset, initialization software must set up the programmable parameters in the memory controller banks registers (OR*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.

13.4.3.3 Intel PC133 and JEDEC-Standard SDRAM Interface Commands

The SDRAM machine performs all accesses to SDRAM by using Intel PC133 and JEDEC-standard SDRAM interface commands. The SDRAM device samples the command and

data inputs on the rising edge of the bus clock. Data at the output of the SDRAM device is sampled on the rising edge of the bus clock.

The following SDRAM interface commands are provided by setting LSDMR[OP] to a non-zero value (LSDMR[OP] = 000 sets normal read/write operation):

Table 13-28. SDRAM Interface Commands

Command (LSDMR[OP])	Description
ACTIVATE (110)	Latches the row address and initiates a memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored with a PRECHARGE command before another ACTIVATE is issued.
MODE-SET (011)	Allows setting of SDRAM options—CAS latency and burst length. CAS latency depends on the SDRAM device used. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, or a page, the local bus memory controller supports only 8-beat bursts for 8-bit and 32-bit port size, or 4-beat bursts for 16-bit 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.

13.4.3.4 Page Hit Checking

The SDRAM machine supports page-mode operation. Each time a page is activated on the SDRAM device, the SDRAM machine stores its address in a page register. The page information, which the user writes to the 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.

13.4.3.5 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.

13.4.3.6 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 13-34 shows how the SDRAM controller shifts the row address down to the lower output address signals during activate and shifts the bank select bits up to the address pins specified by LSDMR[BSMA], supporting page-based interleaving. The lsb of the logical row address (A*n* in Figure 13-34) is aligned with the connected lsb of LAD (bits 29, 30, and 31 for port sizes of 32, 16, and 8 bits, respectively).

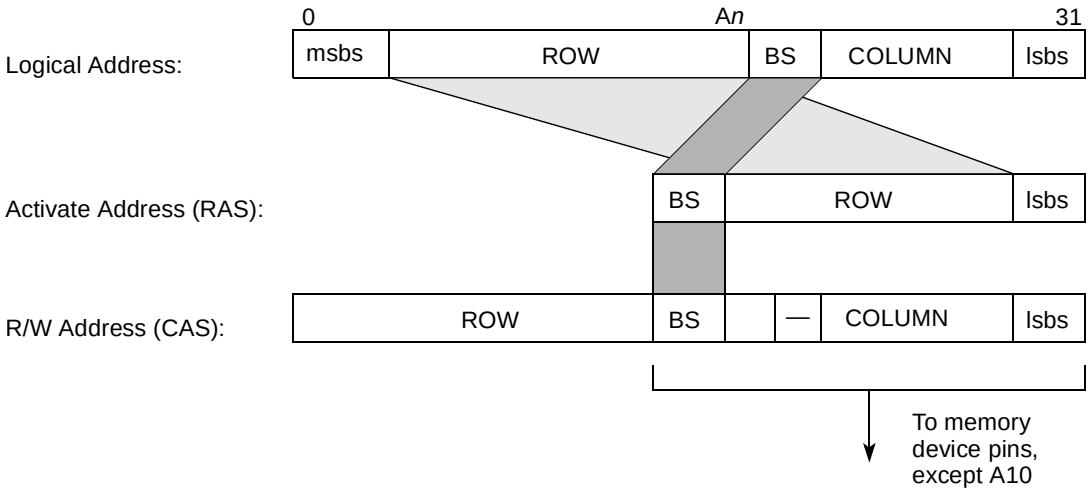


Figure 13-34. SDRAM Address Multiplexing

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.

13.4.3.7 SDRAM Device-Specific Parameters

The software is responsible for setting correct values for device-specific parameters that can be extracted from the device's data sheet. The values are stored in the 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.

13.4.3.7.1 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.

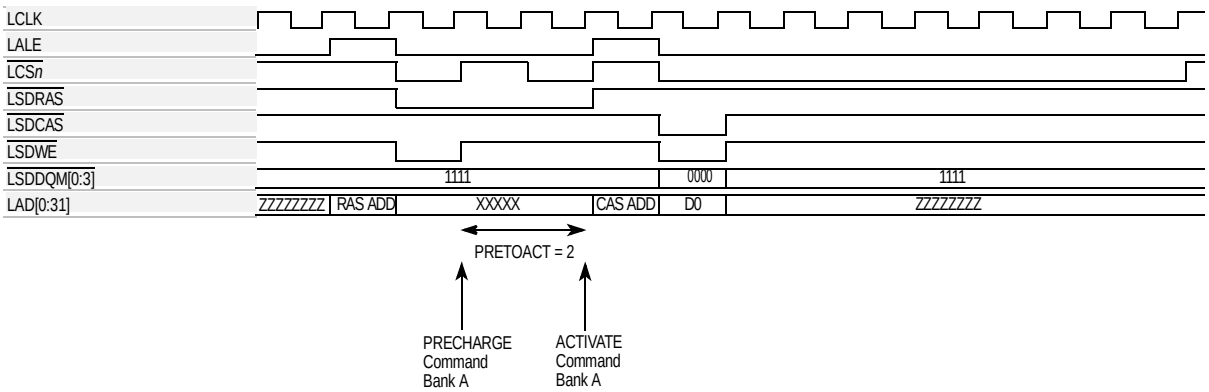


Figure 13-35. PRETOACT = 2 (2 Clock Cycles)

13.4.3.7.2 Activate-to-Read/Write Interval

This parameter, controlled by LSDMR[ACTTORW], defines the earliest timing for a READ/WRITE command after an ACTIVATE command to the same SDRAM bank.

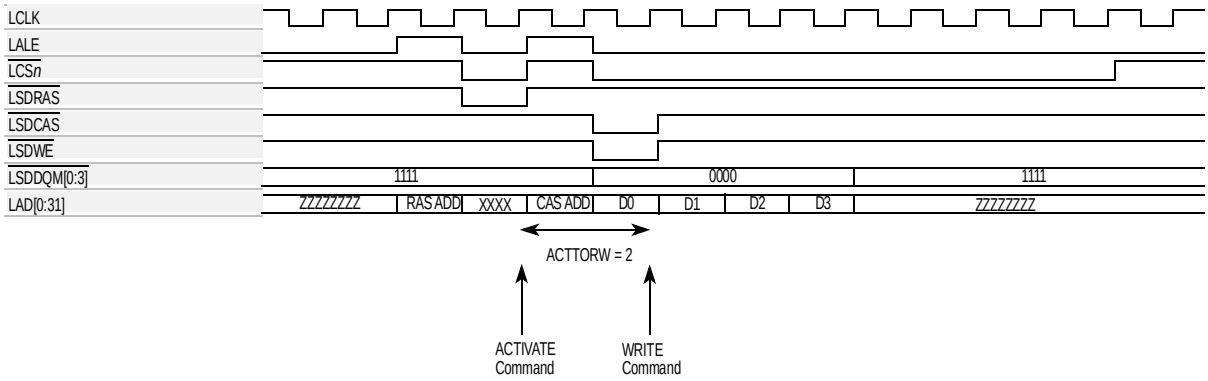


Figure 13-36. ACTTORW = 2 (2 Clock Cycles)

13.4.3.7.3 Column Address to First Data Out—$\overline{\text{CAS}}$ 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.

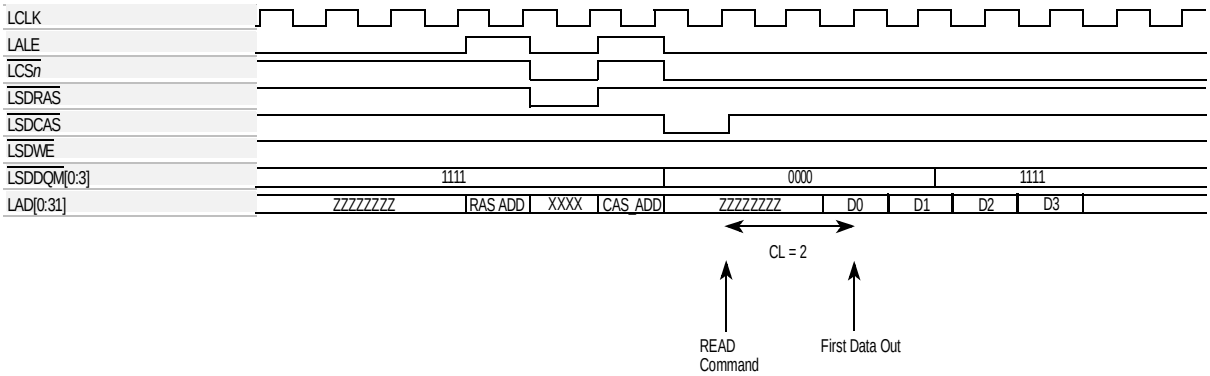


Figure 13-37. CL = 2 (2 Clock Cycles)

13.4.3.7.4 Last Data In to Precharge—Write Recovery

This parameter, controlled by LSDMR[WRC], defines the earliest timing for a PRECHARGE command after the last data was written to the SDRAM.

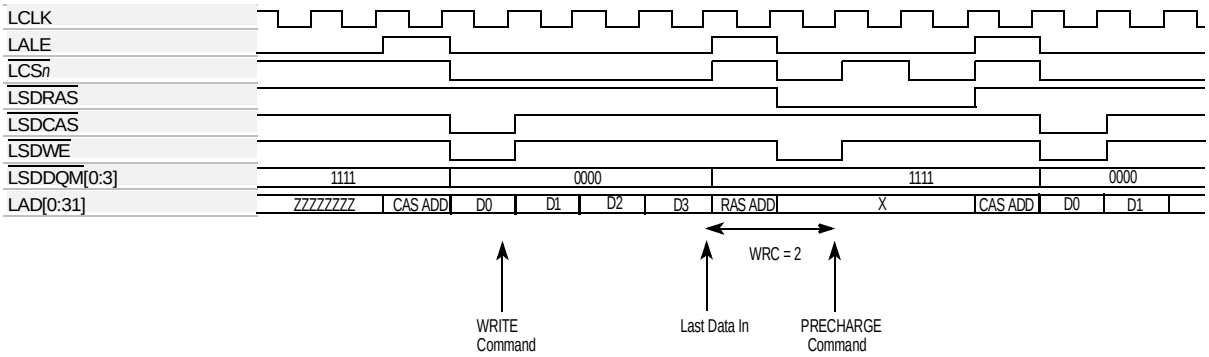


Figure 13-38. WRC = 2 (2 Clock Cycles)

13.4.3.7.5 Refresh Recovery Interval (RFRC)

This parameter, controlled by LSDMR[RFRC], defines the earliest timing for an ACTIVATE or REFRESH command after a REFRESH command to the same SDRAM device.

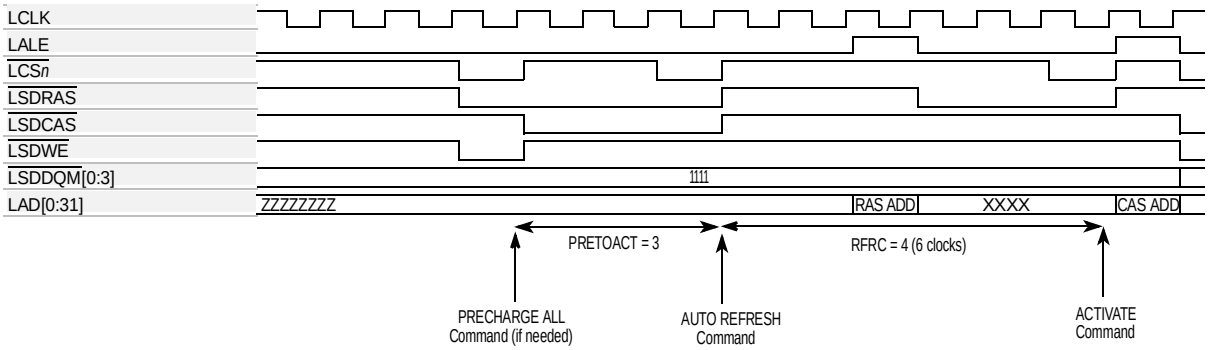


Figure 13-39. RFRC = 4 (6 Clock Cycles)

13.4.3.7.6 External Address and Command Buffers (BUFCMD)

If the additional delay of any buffers placed on the command strobes ($\overline{\text{LSDRAS}}$, $\overline{\text{LSDCAS}}$, $\overline{\text{LSDWE}}$ and LSDA10), is endangering the device setup time, 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.

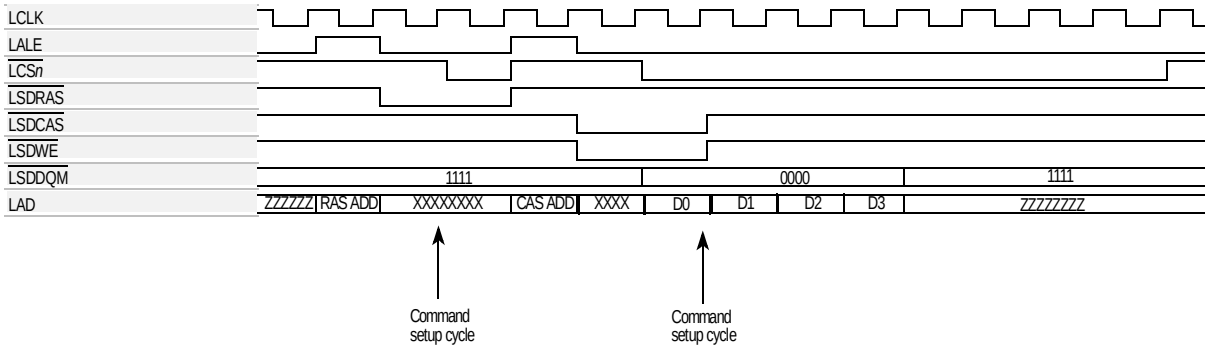


Figure 13-40. BUFCMD = 1, LCRR[BUFCMDC] = 2

13.4.3.8 SDRAM Interface Timing

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

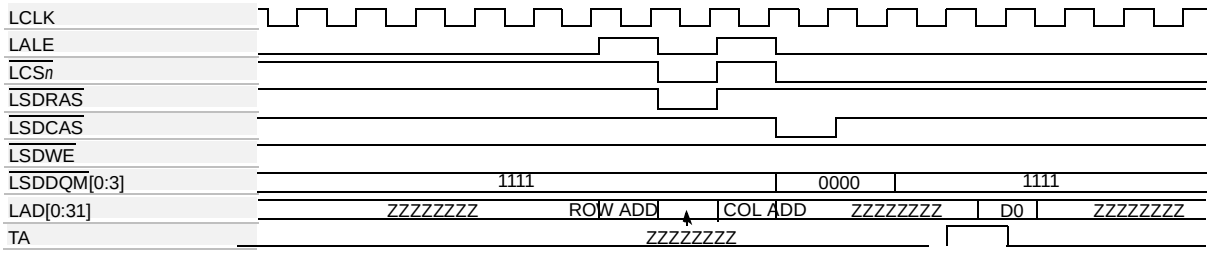


Figure 13-41. SDRAM Single-Beat Read, Page Closed, CL = 3

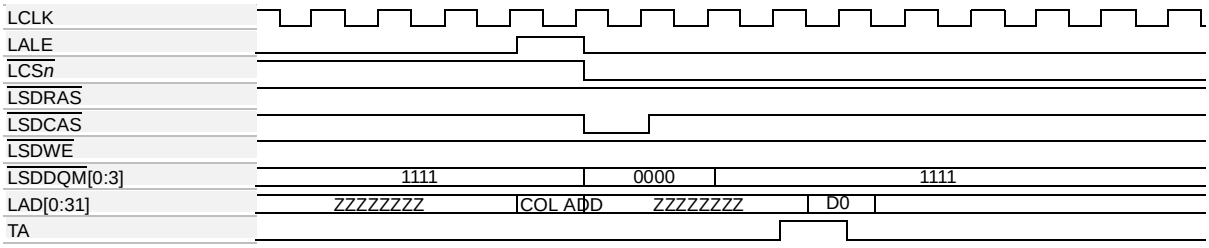


Figure 13-42. SDRAM Single-Beat Read, Page Hit, CL = 3

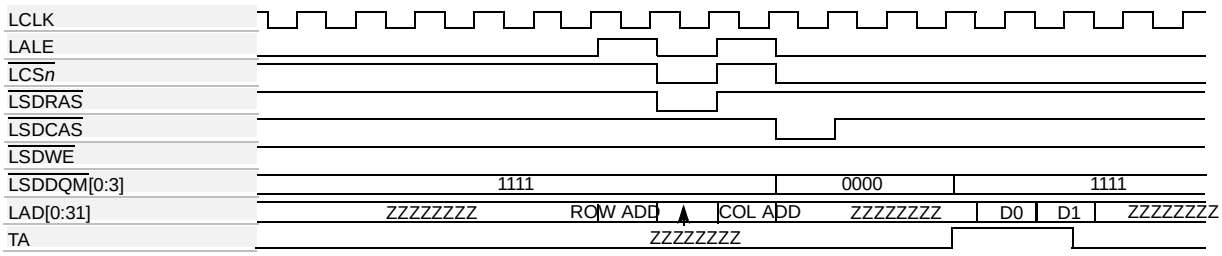


Figure 13-43. SDRAM Two-Beat Burst Read, Page Closed, CL = 3

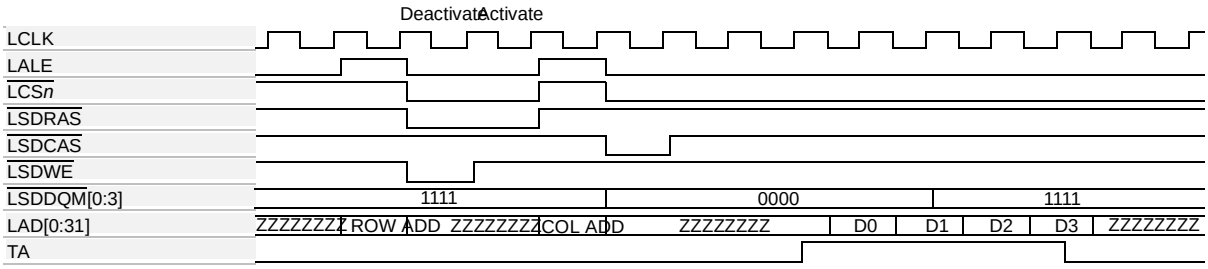


Figure 13-44. SDRAM Four-Beat Burst Read, Page Miss, CL = 3

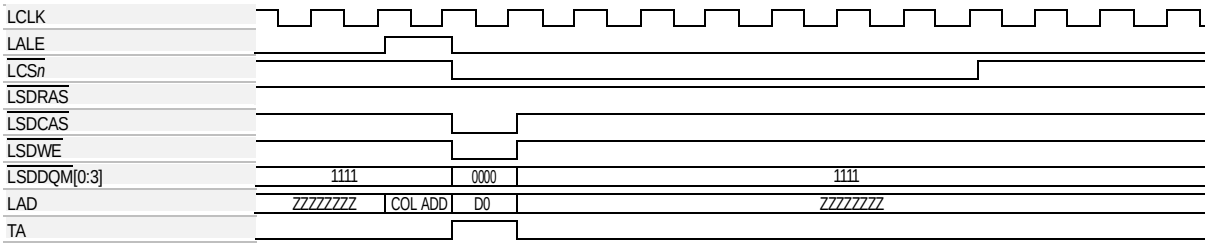


Figure 13-45. SDRAM Single-Beat Write, Page Hit.

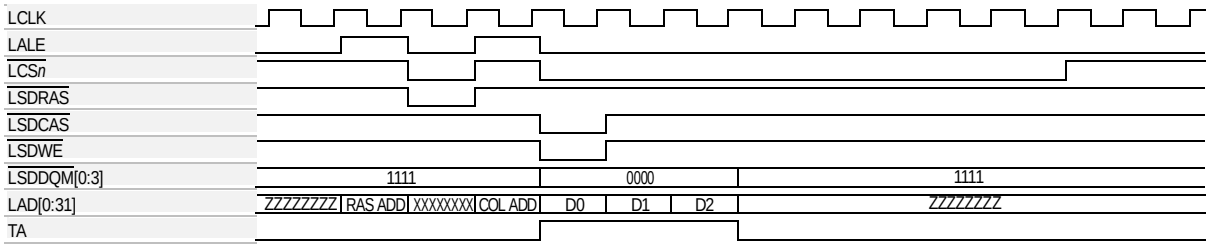


Figure 13-46. SDRAM Three-Beat Write, Page Closed

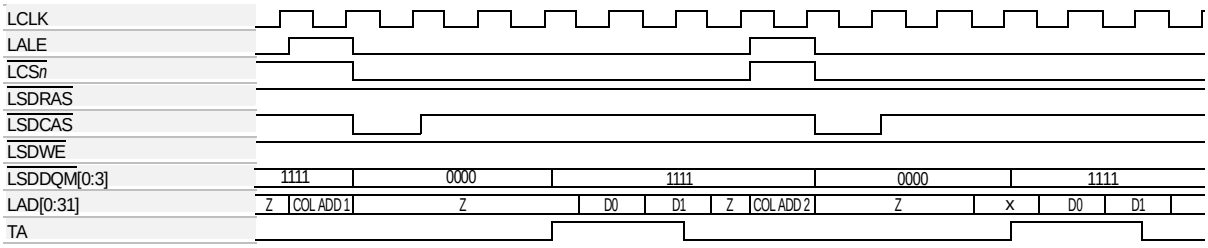


Figure 13-47. SDRAM Read-after-Read Pipelined, Page Hit, CL = 3

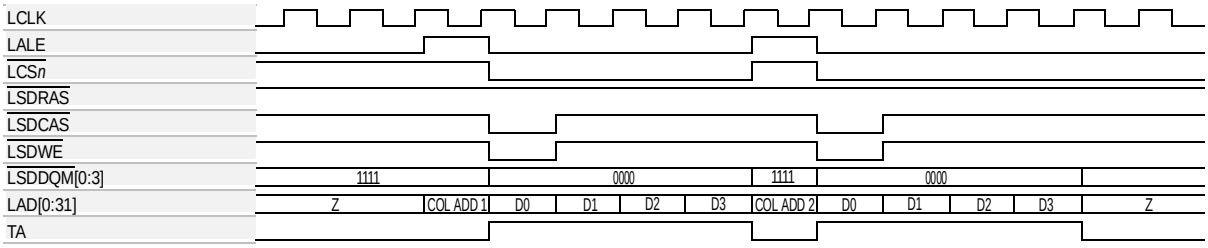


Figure 13-48. SDRAM Write-after-Write Pipelined, Page Hit

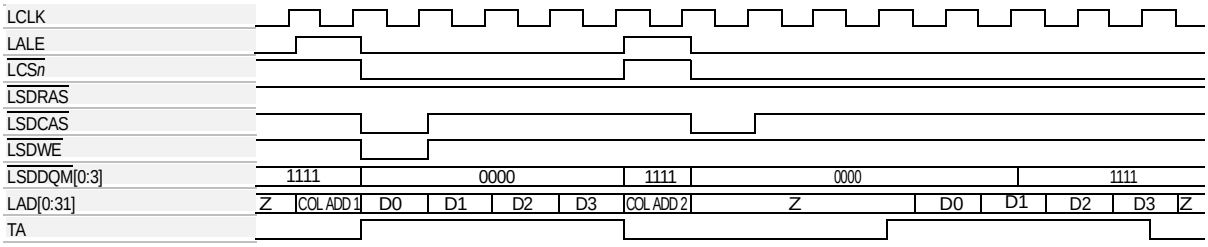


Figure 13-49. SDRAM Read-after-Write Pipelined, Page Hit

13.4.3.9 SDRAM Read/Write Transactions

The SDRAM interface supports read and write transactions of between 1 and 8 data beats for transaction sizes ranging from 1 to 32 bytes. A full burst is performed for each transaction, with the burst length dependent on the port size. A maximum burst of 8 beats is used for an 8-bit or 32-bit port size, while a maximum burst of 4 beats is used for a 16-bit port size, as programmed in LSDMR[BL]. For reads that require less than the full burst length, extraneous data in the burst is ignored and suppressed by the assertion of LSDDQM[0:3]. For writes that require less than the full burst length, the non-targeted addresses are protected by driving corresponding LSDDQM bits high (inactive) on the irrelevant cycles of the burst. However, system performance is not compromised because, if a new transaction is pending, the SDRAM controller begins executing it immediately, effectively terminating the burst early.

13.4.3.10 SDRAM MODE-SET Command Timing

The LBC transfers mode register data (CAS latency and burst length) stored in the LSDMR register to the SDRAM device by issuing the MODE-SET command, as shown in Figure 13-50. In this case, the latched address carries the mode bits for the command.

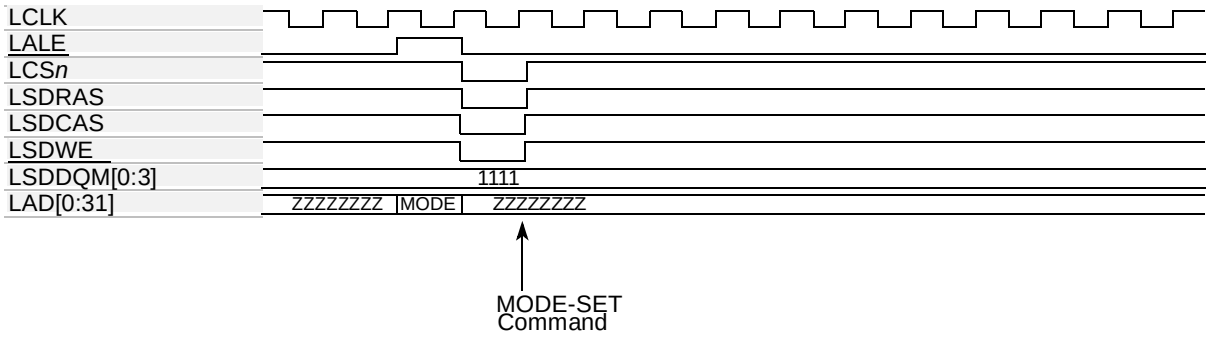


Figure 13-50. SDRAM MODE-SET Command

13.4.3.11 SDRAM Refresh

The memory controller supplies AUTO-REFRESH commands to any connected SDRAM device according to the interval specified in LSRT (and prescaled by MRTPR[PTP]). This represents the time period required between refreshes. The values of LSRT and MRTPR depend on the specific SDRAM devices used and the system clock frequency of the LBC. This value should allow for a potential collision between memory accesses and refresh cycles. The period of the refresh interval must be greater than the access time to ensure that read and write operations complete successfully.

There are two levels of refresh request priority—low and high. The low priority request is generated as soon as the refresh timer expires; this request is granted only if no other requests to the memory controller are pending. If the request is not granted (memory controller is busy) and the refresh timer expires two more times, the request becomes high priority and is served when the current memory controller operation finishes.

13.4.3.11.1 SDRAM Refresh Timing

The SDRAM memory controller implements bank staggering for the auto refresh function. This reduces instantaneous current consumption for memory refresh operations.

After a refresh request is granted, the memory controller begins issuing an AUTO-REFRESH command to each device associated with the refresh timer. After a refresh command is issued to an SDRAM device, the memory controller waits for the number of bus clock cycles programmed in the SDRAM machine's mode register (LSDMR[RFCR]) before issuing any subsequent ACTIVATE command to the same device. To avoid violating SDRAM device timing constraints, the user should ensure that the refresh request interval, defined by LSRT and MRTPR, is greater than the refresh recovery interval, defined by LSDMR[RFCR].

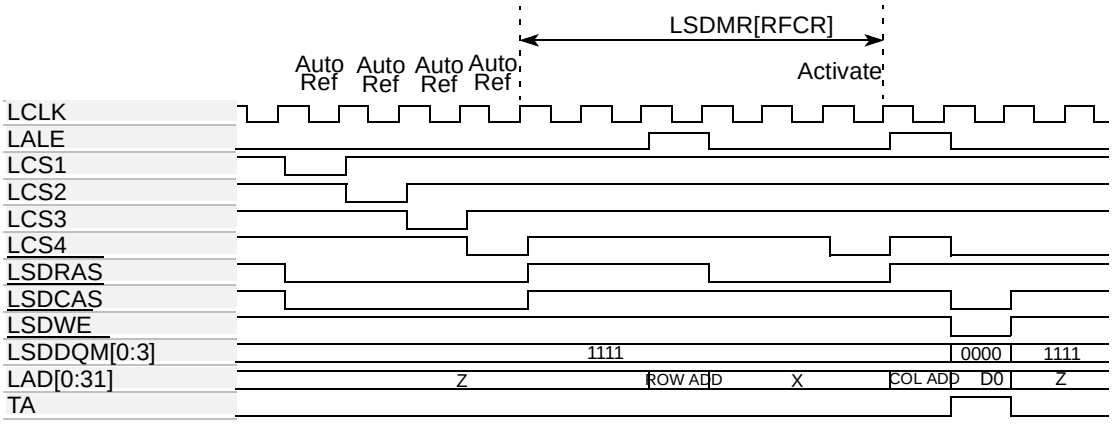


Figure 13-51. SDRAM Bank-Staggered Auto-Refresh Timing

13.4.4 User-Programmable Machines (UPMs)

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 13-52 shows the basic operation of each UPM.

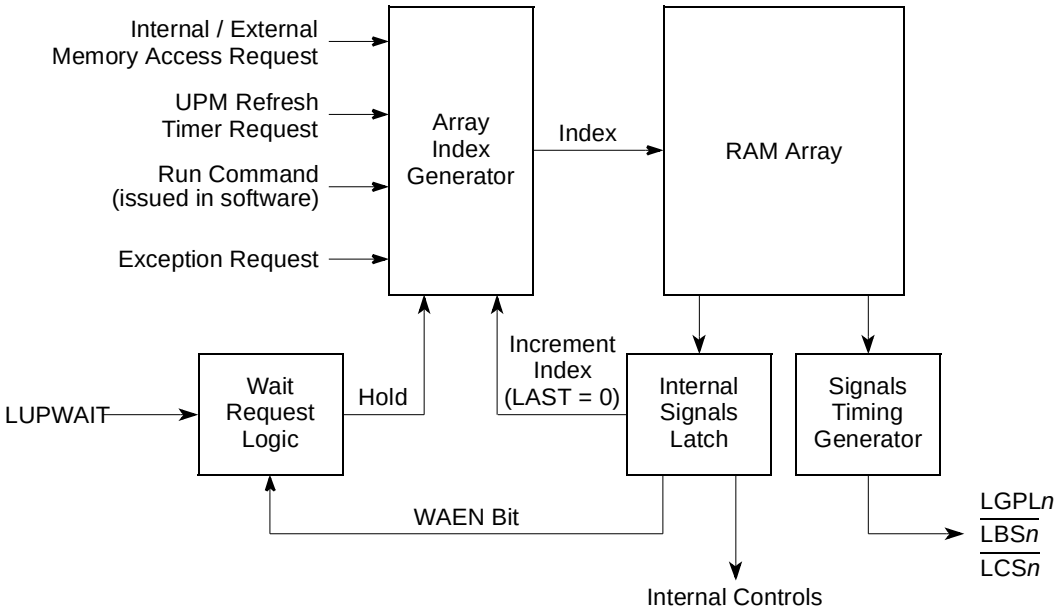


Figure 13-52. User-Programmable Machine Functional Block Diagram

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.

13.4.4.1 UPM Requests

A special pattern location in the RAM array is associated with each of the possible UPM requests. An internal device's request for a memory access initiates one of the following patterns (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 13-53 and Table 13-29 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.

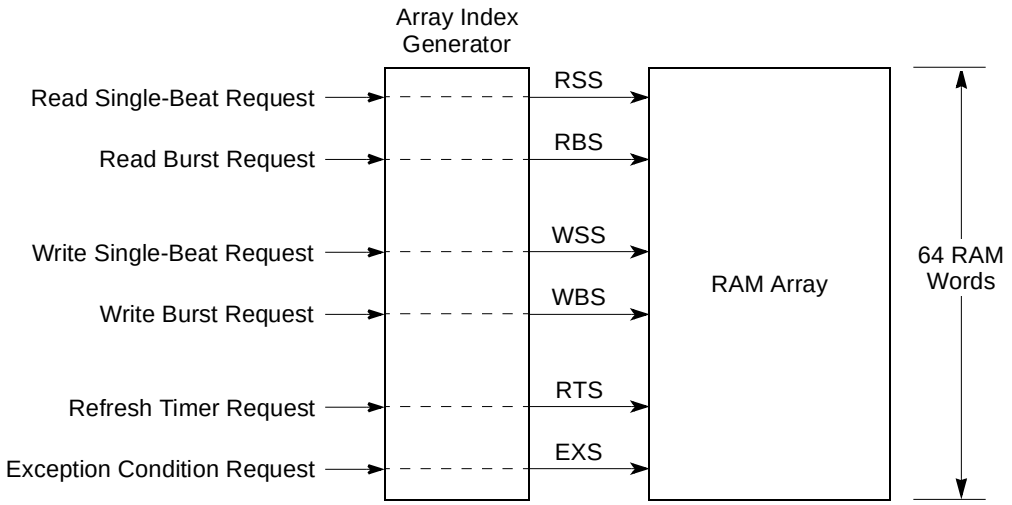


Figure 13-53. RAM Array Indexing

Table 13-29. UPM Routines Start Addresses

UPM Routine	Routine Start Address
Read single-beat (RSS)	0x00
Read burst (RBS)	0x08
Write single-beat (WSS)	0x18
Write burst (WBS)	0x20
Refresh timer (RTS)	0x30
Exception condition (EXS)	0x3C

13.4.4.1.1 Memory Access Requests

The user must ensure that the UPM is appropriately initialized before a request occurs.

The UPM supports two types of memory reads and writes:

- A single-beat transfer transfers one operand consisting of up to a single word (dependent on port size). A single-beat cycle starts with one transfer start and ends with one transfer acknowledge.
- A burst transfer transfers exactly 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.

13.4.4.1.2 UPM Refresh Timer Requests

Each UPM contains a refresh timer that can be programmed to generate refresh service requests of a particular pattern in the RAM array. Figure 13-54 shows the clock division hardware associated with memory refresh timer request generation. The UPM refresh timer register (LURT) defines the period for the timers associated with all three UPMs.

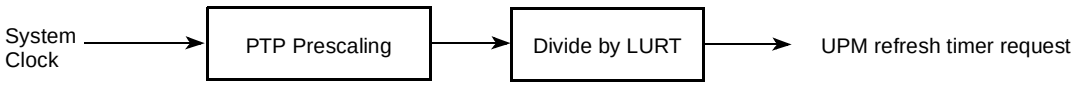


Figure 13-54. Memory Refresh Timer Request Block Diagram

By default, all local bus refreshes are performed using the refresh pattern of UPMA. This means that if refresh is required, MAMR[RFEN] must be set. It also means that only one refresh routine should be programmed and be placed in UPMA, which serves as the refresh executor. Any banks assigned to a UPM are provided with the refresh pattern if the RFEN bit of the corresponding UPM is set. UPMA assigned banks, therefore, always receive refresh services when MAMR[RFEN] is set, while UPMB and UPMC assigned banks also receive (the same) refresh services if the corresponding 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.

13.4.4.1.3 Software Requests—RUN Command

Software can start a request to the UPM by issuing a RUN command to the UPM. Some memory devices have their own signal handshaking protocol to put them into special modes, such as self-refresh mode. Other memory devices require special commands to be issued on their control signals, such as for SDRAM initialization.

For these special cycles, the user creates a special RAM pattern that can be stored in any unused areas in the UPM RAM. Then 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.

13.4.4.1.4 Exception Requests

When the LBC under UPM control initiates an access to a memory device and an exception occurs (bus monitor time-out), the UPM provides a mechanism by which memory control signals can meet the device's timing requirements without losing data. The mechanism is the exception pattern that defines how the UPM negates its signals in a controlled manner.

13.4.4.2 Programming the UPMs

The UPM is a micro sequencer that requires microinstructions or RAM words to generate signal timings for different memory cycles. Follow these steps to program 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).

13.4.4.3 UPM Signal Timing

RAM word fields specify the value of the various external signals at a granularity of up to four values for each bus clock cycle. The signal timing generator causes external signals to behave according to timing specified in the current RAM word. For LCRR[CLKDIV] = 4 or 8, each bit in the RAM word relating to $\overline{\text{LCS}n}$ 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 13-55 and Figure 13-56. 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.

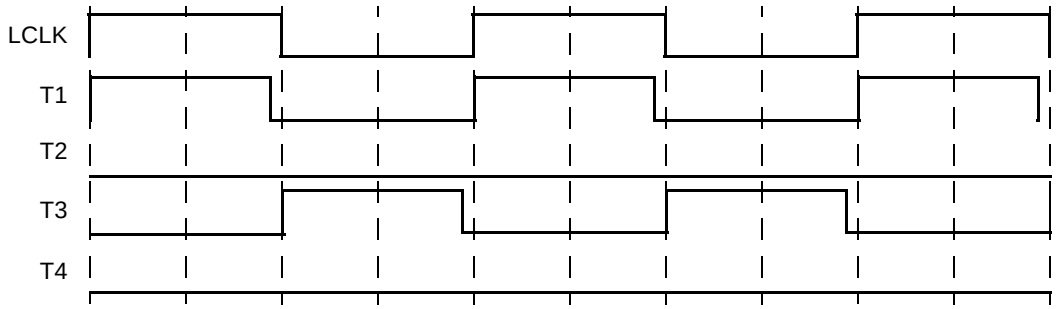


Figure 13-55. UPM Clock Scheme for LCRR[CLKDIV] = 2

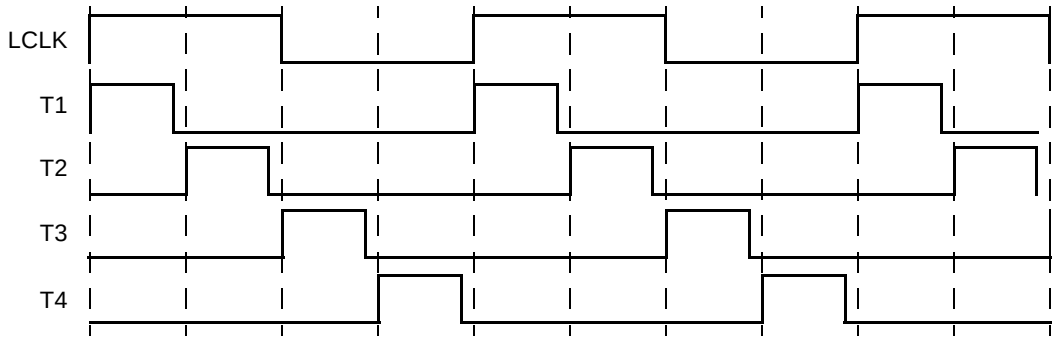


Figure 13-56. UPM Clock Scheme for LCRR[CLKDIV] = 4 or 8

13.4.4.4 RAM Array

The RAM array for each UPM is 64 locations deep and 32 bits wide, as shown in Figure 13-57. The signals at the bottom of the figure are UPM outputs. The selected $\overline{\text{LCS}n}$ 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.

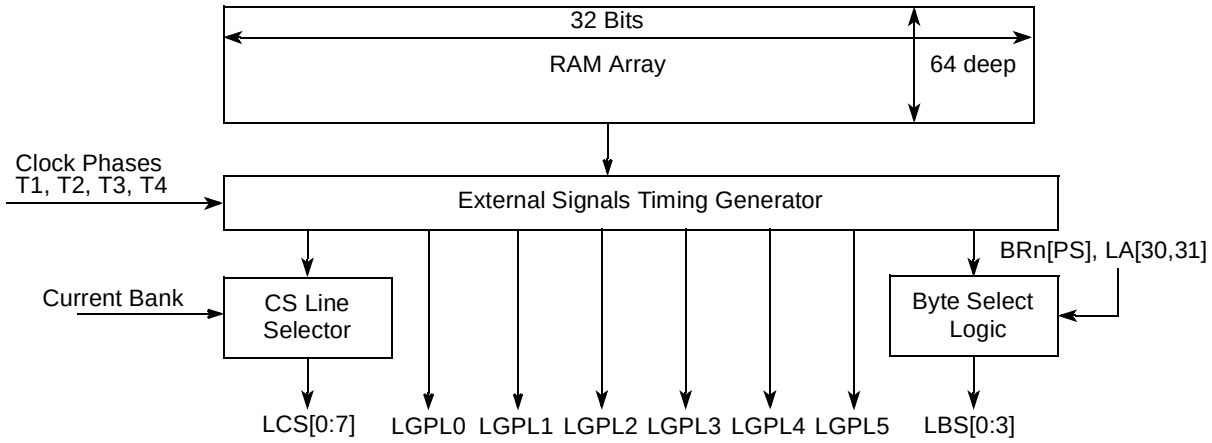


Figure 13-57. RAM Array and Signal Generation

13.4.4.4.1 RAM Words

The RAM word is a 32-bit microinstruction stored in one of 64 locations in the RAM array. It specifies timing for external signals controlled by the UPM. Figure 13-58 shows the RAM word fields. When 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 13-58. RAM Word Field Descriptions

Table 13-30 describes RAM word fields.

Table 13-30. RAM Word Field Descriptions

Bits	Name	Description
0	CST1	Chip select timing 1. Defines the state (0 or 1) of $\overline{\text{LCS}n}$ 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 13-30. 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 13-30. 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.

13.4.4.4.2 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 13-59 shows how UPMs control $\overline{\text{LCS}n}$ signals.

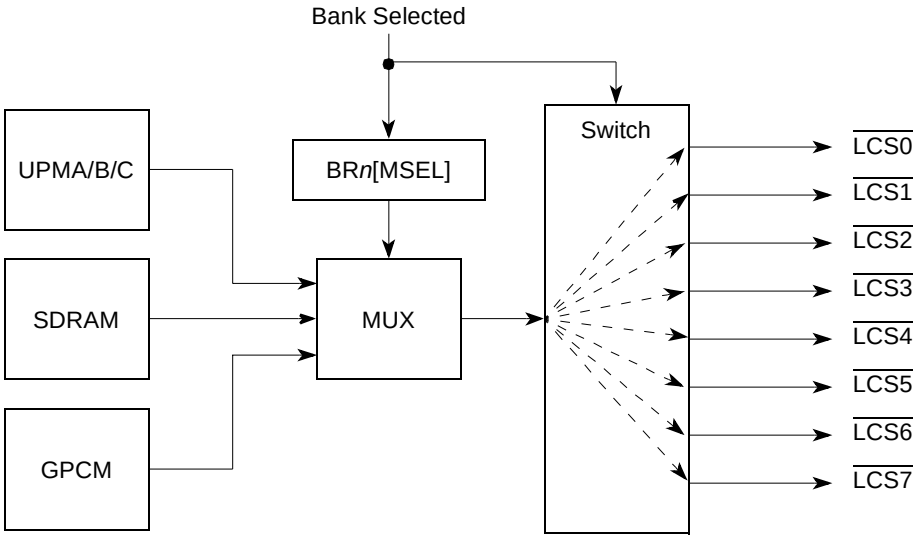


Figure 13-59. $\overline{\text{LCS}n}$ Signal Selection

13.4.4.4.3 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 13-60 shows how UPMs control $\overline{\text{LBS}}$[0:3].

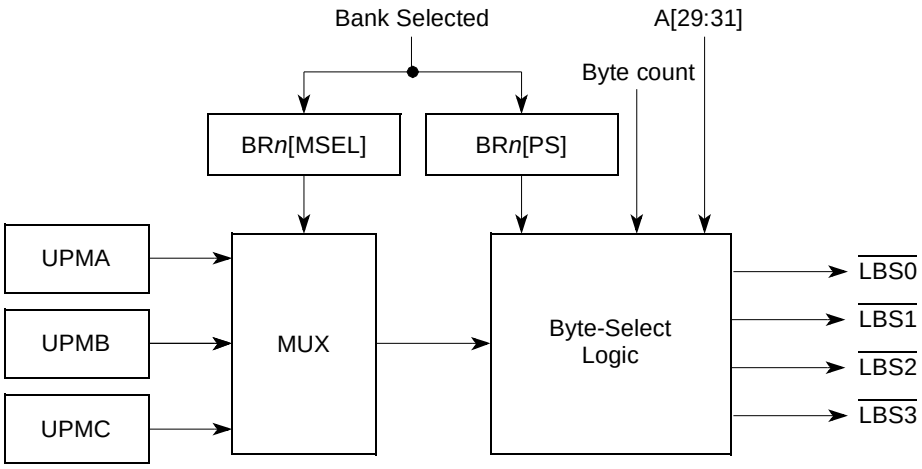


Figure 13-60. $\overline{\text{LBS}}$ Signal Selection

The uppermost byte select ($\overline{\text{LBS0}}$), when asserted, indicates that LAD[0:7] contains valid data during a cycle. Likewise, $\overline{\text{LBS1}}$ indicates that LAD[8:15] contains valid data, $\overline{\text{LBS2}}$ indicates that LAD[16:23] contains valid data, and $\overline{\text{LBS3}}$ indicates that LAD[24:31] contains valid data. For a UPM refresh timer request, all $\overline{\text{LBS}}$[0:3] signals are asserted/negated by the UPM according to the refresh pattern only. Following any internal bus monitor exception, $\overline{\text{LBS}}$[0:3] signals are negated regardless of the exception handling provided by any UPM exception pattern to prevent spurious writes to external RAM.

13.4.4.4.4 General-Purpose Signals (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.

13.4.4.4.5 Loop Control (LOOP)

The LOOP bit in the RAM word specifies the beginning and end of a set of UPM RAM words that are to be repeated. The first time LOOP = 1, the memory controller recognizes it as a loop start word and loads the memory loop counter with the corresponding contents of the loop field shown in Table 13-31. The next RAM word for which LOOP = 1 is recognized as a loop end word. When it is reached, the loop counter is decremented by one.

Continued loop execution depends on the loop counter. If the counter is not zero, the next RAM word executed is the loop start word. Otherwise, the next RAM word executed is the one after the

loop end word. Loops can be executed sequentially but cannot be nested. Also, special care must be taken if LAST and LOOP must not be set together.

Table 13-31. 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

13.4.4.4.6 Repeat Execution of Current RAM Word (REDO)

The REDO function is useful for wait-state insertion in a long UPM routine that would otherwise need too many RAM words. Setting the REDO bits of the RAM word to a nonzero value causes the UPM to re-execute the current RAM word up to three more times, as defined in the REDO field of the current RAM word.

Special care must be taken in the following cases:

- When UTA and REDO are set together, TA is asserted the number of times specified by the REDO function.
- When NA and REDO are set together, the address is incremented the number of times specified by the REDO function.
- When LOOP and REDO are set together, the loop mechanism works as usual and the line is repeated according to the REDO function.
- LAST and REDO must not be set together.
- REDO should not be used within the exception routine.

13.4.4.4.7 Address Multiplexing (AMX)

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 13-32 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 13-32. 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.

13.4.4.4.8 Data Valid and Data Sample Control (UTA)

When a read access is handled by the UPM, and the UTA bit is 1 (data is to be sampled by the 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 13-61 shows how data sampling is controlled by the UPM.

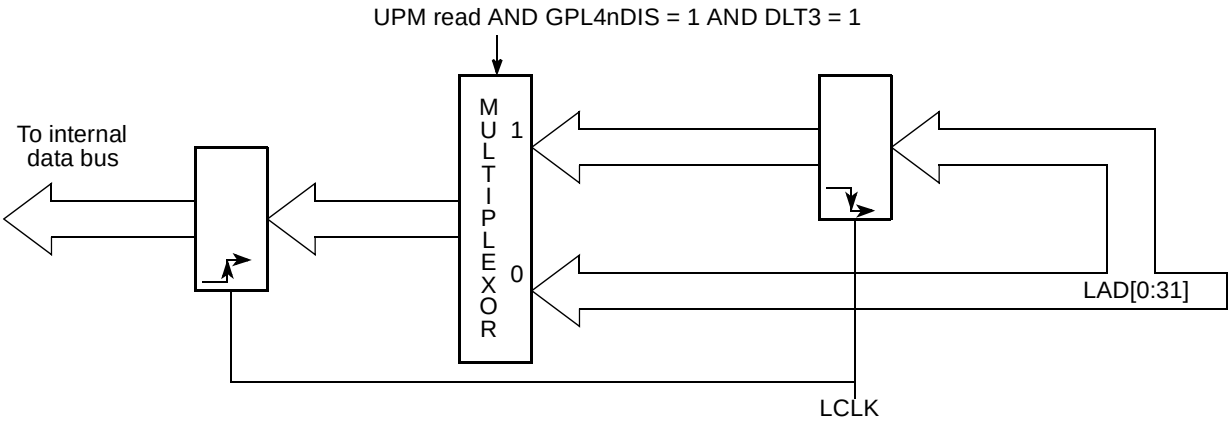


Figure 13-61. UPM Read Access Data Sampling

13.4.4.4.9 $\overline{\text{LGPL}}$[0:5] Signal Negation (LAST)

When the LAST bit is read in a RAM word, the current UPM pattern is terminated at the end of the current cycle. On the next cycle (following LAST) all the UPM signals are negated unconditionally (driven to logic 1), unless there is a back-to-back UPM request pending. In this case, the signal values for the cycle following the one in which the LAST bit was set are taken from the first RAM word of the pending UPM routine.

13.4.4.4.10 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 13-62 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.

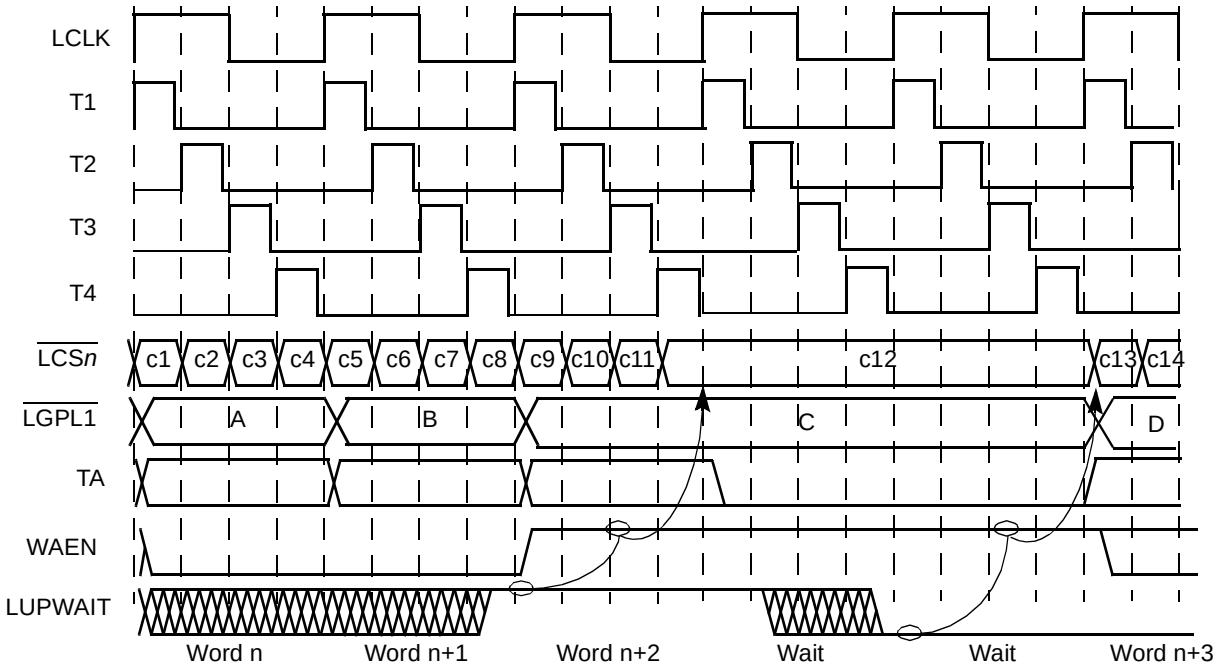


Figure 13-62. Effect of LUPWAIT Signal

13.4.4.5 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.

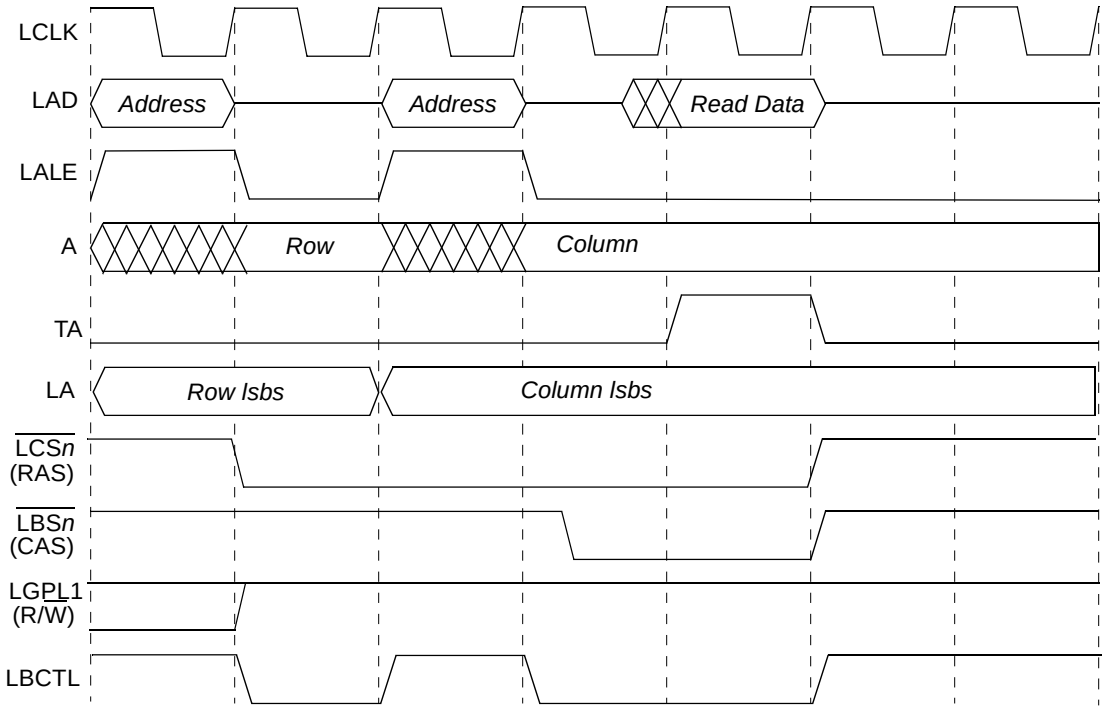
However, programming WAEN = 1 and UTA = 1 in the same RAM word allows UPM to treat LUPWAIT as a synchronous signal, which must meet set-up and hold times in relation to the rising edge of the bus clock. In this case, as soon as UPM samples LUPWAIT negated on the rising edge of the bus clock, it immediately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

13.4.4.6 Extended Hold Time on Read Accesses

Slow memory devices that take a long time to turn off their data bus drivers on read accesses should choose some non-zero combination of 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.

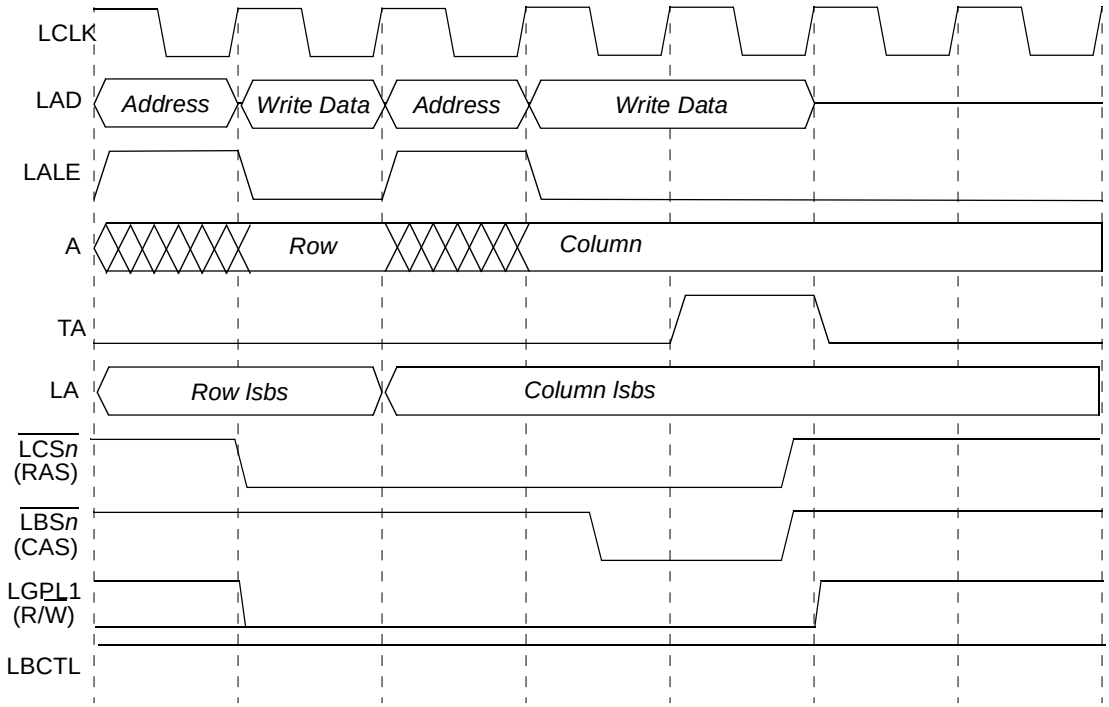
13.4.4.7 Memory System Interface Example Using UPM

Connecting the local bus UPM controller to a DRAM device requires a detailed examination of the timing diagrams representing the possible memory cycles that must be performed when accessing this device. This section describes timing diagrams for various UPM configurations, using fast-page mode DRAM as an example, with LCRR[CLKDIV] = 4 or 8. 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.



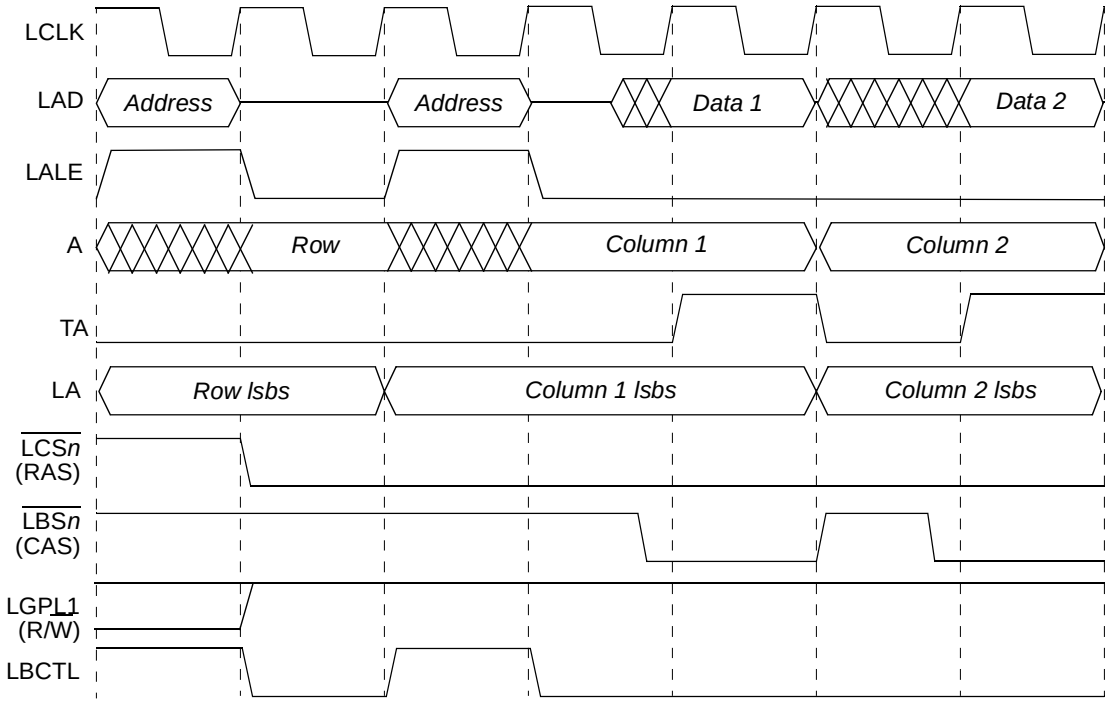
cst1	0	LALE pause (due to change in AMX)	0	0	Bit 0
cst2	0		0	0	Bit 1
cst3	0		0	0	Bit 2
cst4	0		0	0	Bit 3
bst1	1		1	0	Bit 4
bst2	1		0	0	Bit 5
bst3	1		0	0	Bit 6
bst4	1		0	0	Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1	1		1	1	Bit 12
g1t3	1		1	1	Bit 13
g2t1					Bit 14
g2t3			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

Figure 13-63. Single-Beat Read Access to FPM DRAM



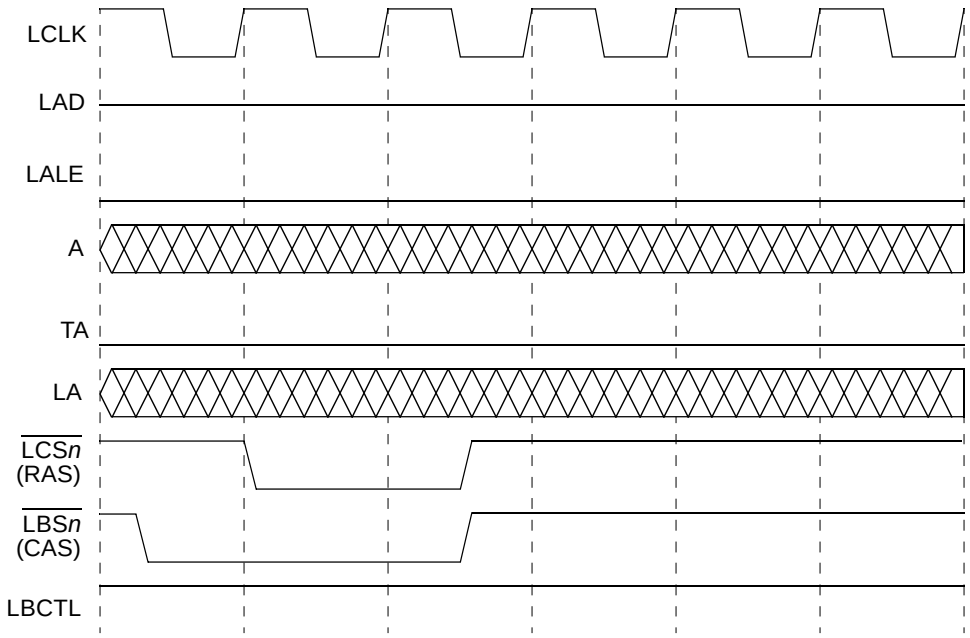
cst1	0	LALE pause (due to change in AMX)	0	0	Bit 0
cst2	0		0	0	Bit 1
cst3	0		0	0	Bit 2
cst4	0		0	1	Bit 3
bst1	1		1	0	Bit 4
bst2	1		1	0	Bit 5
bst3	1		0	0	Bit 6
bst4	1		0	1	Bit 7
g0l0					Bit 8
g0l1					Bit 9
g0h0					Bit 10
g0h1					Bit 11
g1t1	0		0	0	Bit 12
g1t3	0		0	0	Bit 13
g2t1					Bit 14
g2t3			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

Figure 13-64. Single-Beat Write Access to FPM DRAM



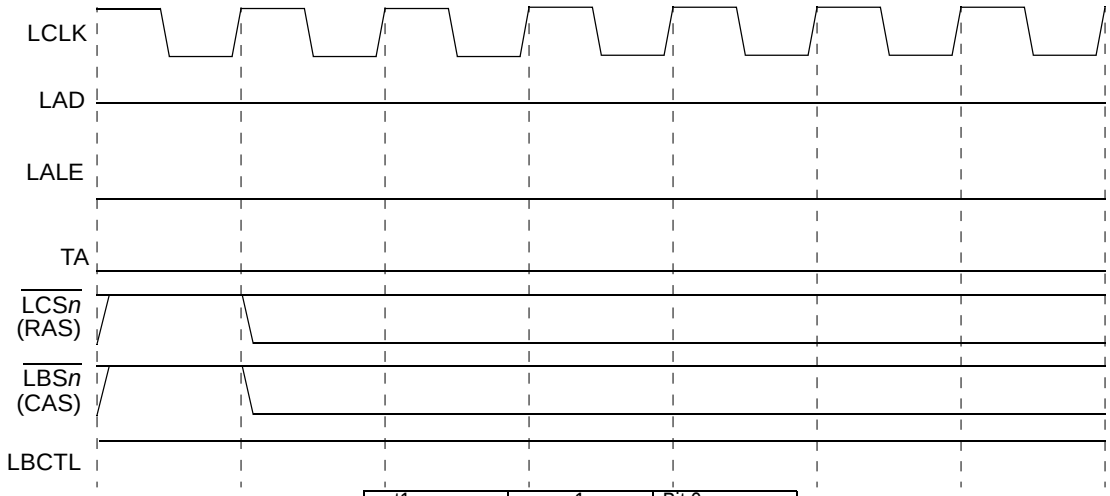
cst1	0	LALE pause (due to change in AMX)	0	0	1	Bit 0
cst2	0		0	0	1	Bit 1
cst3	0		0	0	1	Bit 2
cst4	0		0	0	1	Bit 3
bst1	1		1	0	1	Bit 4
bst2	1		1	0	1	Bit 5
bst3	1		1	0	1	Bit 6
bst4	1		0	0	1	Bit 7
g0l0						Bit 8
g0l1						Bit 9
g0h0						Bit 10
g0h1						Bit 11
g1t1	1		1	1	1	Bit 12
g1t3	1		1	1	1	Bit 13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

Figure 13-65. 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 13-66. Refresh Cycle (CBR) to FPM DRAM



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

Figure 13-67. Exception Cycle

13.5 Initialization/Application Information

13.5.1 Interfacing to Peripherals

13.5.1.1 Multiplexed Address/Data Bus and Unmultiplexed Address Signals

To save pins on the local bus, address and data are multiplexed onto the same 32 bit bus. An external latch is needed to 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.

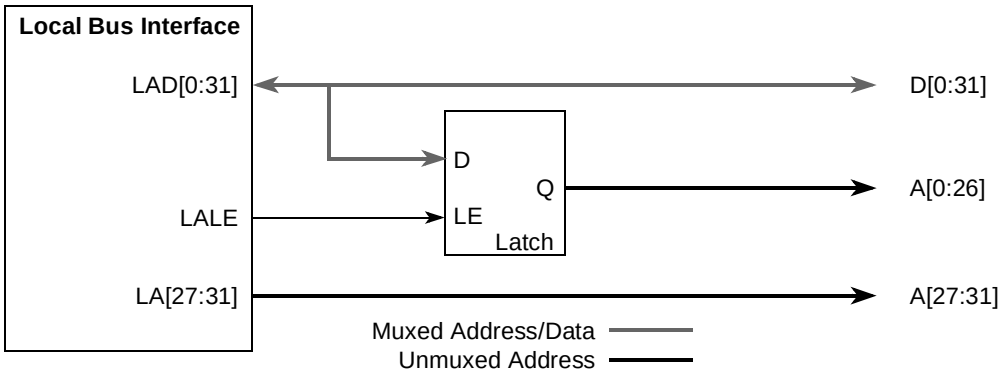


Figure 13-68. Multiplexed Address/Data Bus

13.5.1.2 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 13-69.

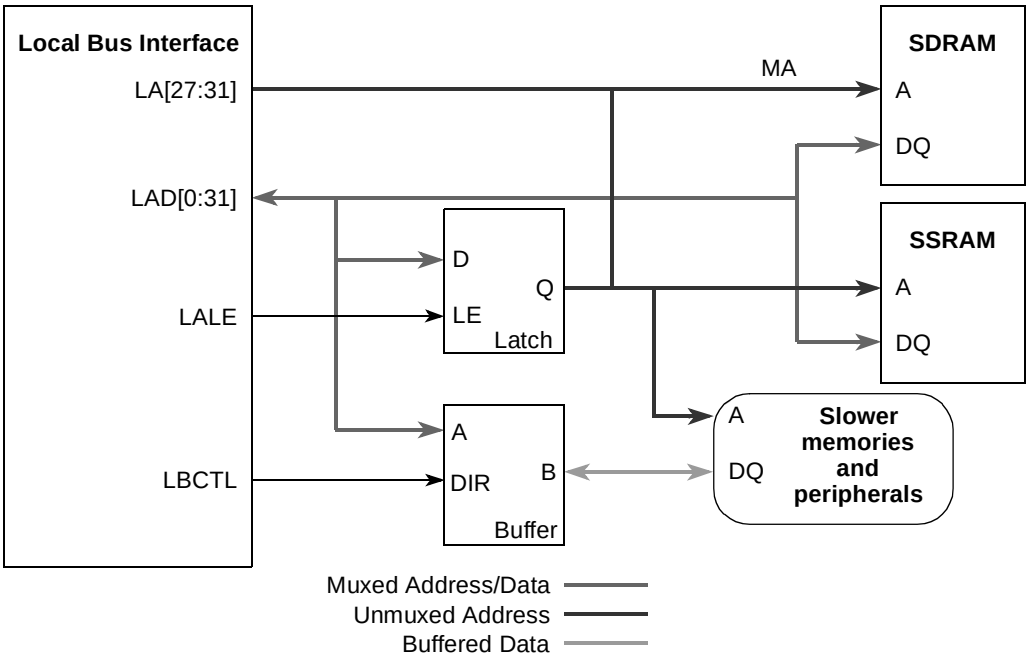


Figure 13-69. Local Bus Peripheral Hierarchy

The multiplexed address/data bus sees the capacitive loading of the data pins of the fast SDRAMs or synchronous SRAMs plus one load for an address latch plus one load for a buffer to the slow memories. The loadings of all other memories and peripherals are hidden behind the buffer and the latch. The system designer needs to investigate the loading scenario and ensure that I/O timings can be met with the loading determined by the connected components.

13.5.1.3 Peripheral Hierarchy on the Local Bus for Very High Bus Speeds

To achieve the highest possible bus speeds on the local bus, it is recommended to reduce the number of devices connected directly to the local bus even further. For those cases probably only one bank of synchronous SRAMs or SDRAMs should be used and instead of using a separate latch and a separate bus transceiver, a bus demultiplexer combining those two functions into one device should be used. Figure 13-70 shows an example of such a hierarchy. This section is only a guideline and the board designer must simulate the electric characteristics of his scenario to determine the maximum operating frequency.

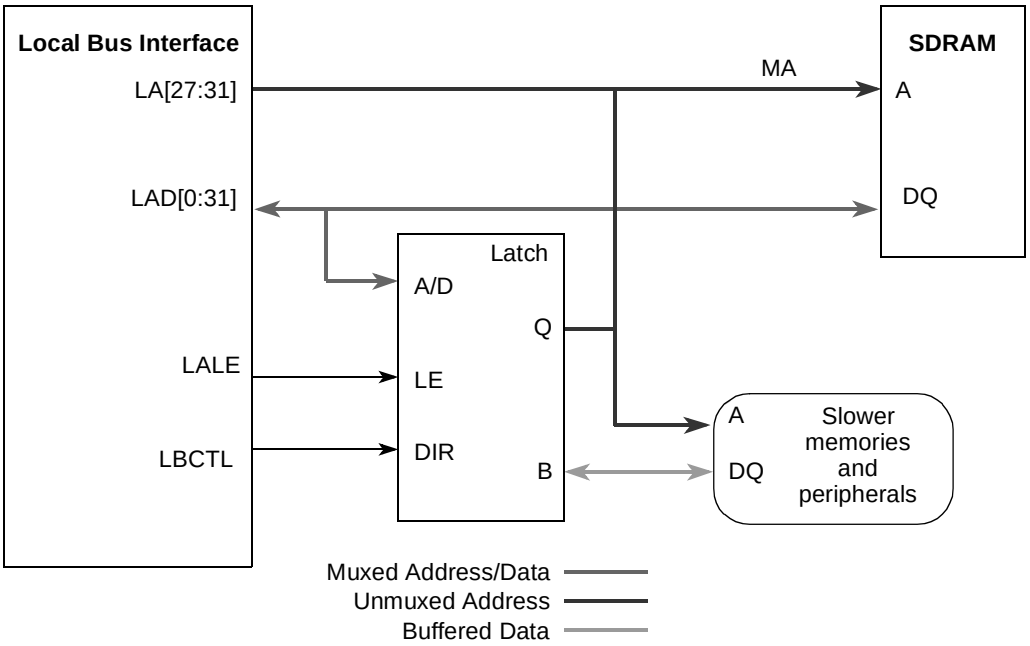


Figure 13-70. Local Bus Peripheral Hierarchy for Very High Bus Speeds

13.5.1.4 GPCM Timings

In case a system contains a memory hierarchy with high speed synchronous memories (SDRAM, synchronous SRAM) and lower speed asynchronous memories (for example, FLASH EPROM and peripherals) the GPCM-controlled memories should be decoupled by buffers to reduce capacitive loading on the bus. Those buffers have to be taken into account for the timing calculations.

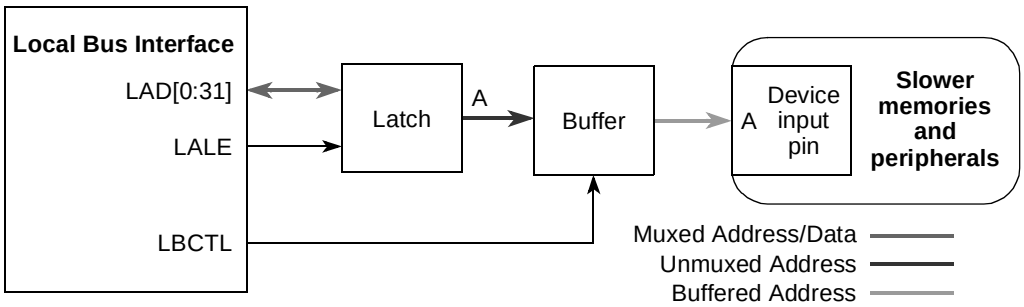


Figure 13-71. GPCM Address Timings

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.

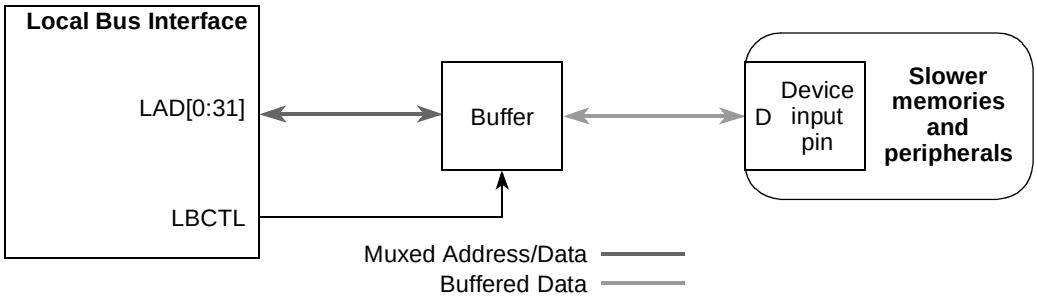


Figure 13-72. GPCM Data timings

13.5.2 Bus Turnaround

Because the local bus uses multiplexed address and data, special consideration must be given to avoid bus contention at bus turnaround. The following cases must be examined:

- Address phase after previous read
- Read data phase after address phase
- Read-modify-write cycle for parity protected memory banks
- UPM cycles with additional address phases

The bus does not change direction for the following cases so they need no special attention:

- Continued burst after the first beat
- Write data phase after address phase
- Address phase after previous write

13.5.2.1 Address Phase After Previous Read

During a read cycle, the memory/peripheral drives the bus and the bus transceiver drives LAD. After the data has been sampled, the output drivers of the external device must be disabled. This can take some time; for slow devices the EHTR feature of the GPCM or the programmability of the UPM should be used to guarantee that those devices have stopped driving the bus when the LBC memory controller ends the bus cycle.

In this case, after the previous cycle ends, LBCTL goes high and changes the direction of the bus transceiver. The LBC then inserts a bus turnaround cycle to avoid contention. The external device has now already placed its data signals in high impedance and no bus contention will occur.

13.5.2.2 Read Data Phase After Address Phase

During the address phase, LAD actively drives the address and LBCTL is high, driving 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.

13.5.2.3 Read-Modify-Write Cycle for Parity Protected Memory Banks

Principally, a read-modify-write cycle is a read cycle immediately followed by a write cycle. Because the write cycle will have a new address phase in any case, this basically is the same case as an address phase after a previous read.

13.5.2.4 UPM Cycles with Additional Address Phases

The flexibility of the UPM allows the user to insert additional address phases during read cycles by changing the AMX field, therefore turning around the bus during one pattern. The LBC automatically inserts a single bus turnaround cycle if the bus (LAD) was previously high impedance for any reason, such as a read, before LALE is driven and LAD is driven with the new address. The turnaround cycle is not inserted on a write, because the bus was already driven to begin with.

However, bus contention could potentially still occur on the far side of a bus transceiver. It is the responsibility of the designer of the UPM pattern to guarantee that enough idle cycles are inserted in the UPM pattern to avoid this.

13.5.3 Interface to Different Port-Size Devices

The 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 13-73 shows the device connections on the data bus.

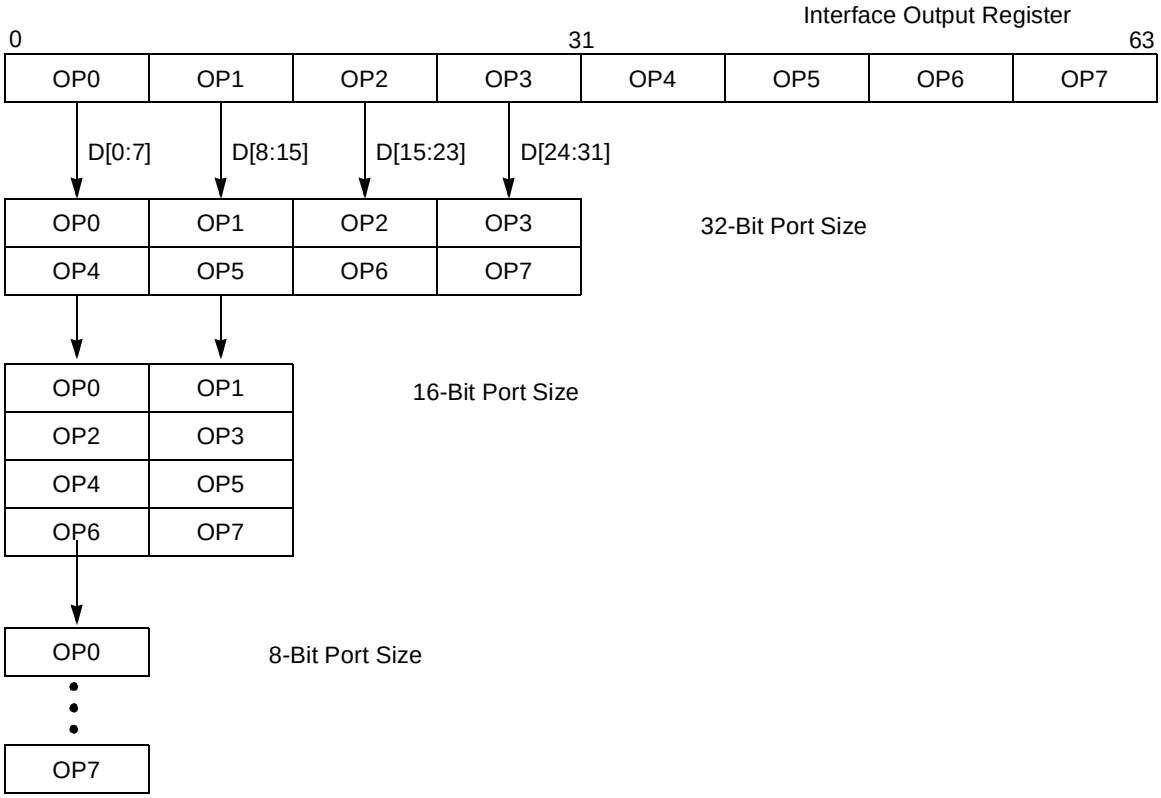


Figure 13-73. Interface to Different Port-Size Devices

Table 13-33 lists the bytes required on the data bus for read cycles.

Table 13-33. Data Bus Requirements For Read Cycle

Transfer Size	Address State [1] A[29:31]	Port Size/Data Bus Assignments						
		32-Bit				16-Bit		8-Bit
		0–7	8–15	16–23	24–31	0–7	8–15	0–7
Byte	000	OP0[2]	—[3]	—	—	OP0	—	OP0
	001	—	OP1	—	—	—	OP1	OP1
	010	—	—	OP2	—	OP2	—	OP2
	011	—	—	—	OP3	—	OP3	OP3
	100	OP4	—	—	—	OP4	—	OP4
	101	—	OP5	—	—	—	OP5	OP5
	110	—	—	OP6	—	OP6	—	OP6
	111	—	—	—	OP7	—	OP7	OP7

Table 13-33. Data Bus Requirements For Read Cycle (continued)

Transfer Size	Address State [1] A[29:31]	Port Size/Data Bus Assignments						
		32-Bit				16-Bit		8-Bit
		0–7	8–15	16–23	24–31	0–7	8–15	0–7
Half Word	000	OP0	OP1	—	—	OP0	OP1	OP0
	001	—	OP1	OP2	—	—	OP1	OP1
	010	—	—	OP2	OP3	OP2	OP3	OP2
	100	OP4	OP5	—	—	OP4	OP5	OP4
	101	—	OP5	OP6	—	—	OP5	OP5
	110	—	—	OP6	OP7	OP6	OP7	OP6
Word	000	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	100	OP4	OP5	OP6	OP7	OP4	OP5	OP4

1 Address state is the calculated address for port size.

2 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.

13.5.4 Interfacing to SDRAM

The following subsections provide application information on interfacing to SDRAM.

13.5.4.1 Basic SDRAM Capabilities of the Local Bus

The LBC provides one SDRAM machine for the local bus. Although there is only one machine, multiple chip selects ($\overline{\text{LCS}n}$) 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 13-51 in Section 13.4.3.11.1, "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.

13.5.4.2 Maximum Amount of SDRAM Supported

Table 13-34 summarizes information based on SDRAM data sheets supplied by Micron.

Table 13-34. Micron SDRAM Devices

SDRAM Device	64 Mbit				128 Mbit				256 Mbit				512 Mbit			
I/O Port	x4	x8	x16	x32	x4	x8	x16	x32	x4	x8	x16	x32	x4	x8	x16	x32
Bank	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Row	12	12	12	11	12	12	12	12	13	13	13	13	13	13	13	TBD
Column	10	9	8	8	11	10	9	8	11	10	9	8	12	11	10	TBD

The data port size is programmable, but the following examples use all 32 bits of the local bus. The 32-bit port size requires four SDRAM devices (with 8-bit I/O ports) connected in parallel to a single chip select. If 128-Mbit devices are used, one chip select provides 128-Mbit/device x 4 devices = 64 Mbytes. If 4 chip selects are programmed for SDRAM use, the result is 64 Mbytes x 4 = 256 Mbyte. If 256-Mbit SDRAM devices are used, the total available memory is 512 Mbyte. Consequently 512-Mbit devices allow for 1 Gbyte.

Although there is no technical difficulty in supporting multiple chip select configurations, in practice, the user may want to maximize the amount of SDRAM assigned to each chip select to minimize cost.

13.5.4.3 SDRAM Machine Limitations

This section describes limitations of the local bus SDRAM machine.

13.5.4.3.1 Analysis of Maximum Row Number Due to Bank Select 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.

13.5.4.3.2 Bank Select Signals

Page-based interleaving allows bank signals to be multiplexed to the higher-order address pins to leave room for future upgrades. For example, a user could multiplex the bank select signals to LA[14:15], leaving LA16 to connect to the address pin for a larger memory size.

This allows the system designer to design one board that can be used with a current generation of SDRAM devices and upgraded to the next generation without requiring a new board layout.

13.5.4.3.3 128-Mbyte SDRAM

Figure 13-74 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.

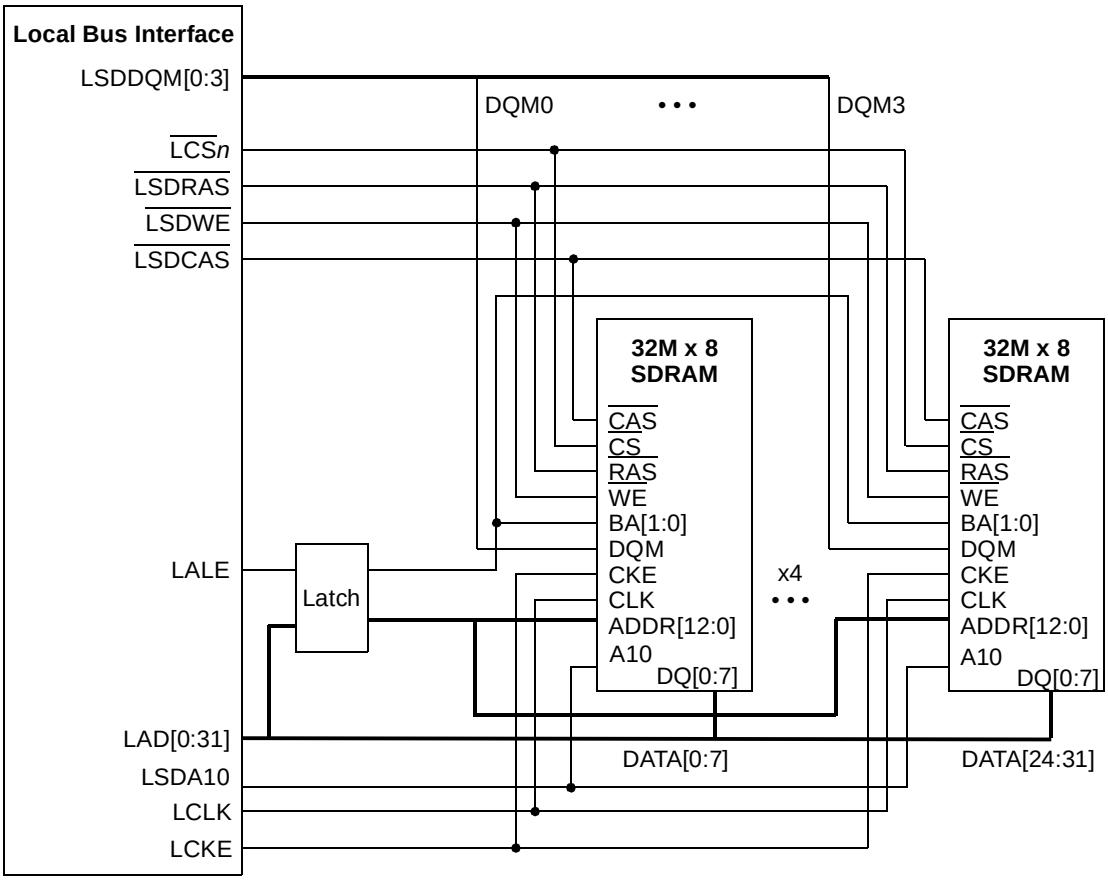


Figure 13-74. 128-Mbyte SDRAM Diagram

Table 13-35 shows details about LAD*n* signal connections for the example in Figure 13-74.

Table 13-35. 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 13-35. 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 13-36.

Table 13-36. 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 13-37.

Table 13-37. 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 13-38 shows the address port configuration during a READ/WRITE command.

Table 13-38. 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 13-39 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 13-39. 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

13.5.4.3.4 256-Mbyte SDRAM

This example uses the same Micron SDRAM as in the previous example, but doubles the number of devices connected and therefore uses two chip selects.

13.5.4.3.5 512-Mbyte SDRAM

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 13-40.

Table 13-40. 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 13-41.

Table 13-41. 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 13-42 shows the address port settings during a READ/WRITE command.

Table 13-42. 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 13-43 shows the register configuration. PSRT and MPTPR are not shown, but they should be programmed according to the specific device's refresh requirements.

Table 13-43. 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 13-43. 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

13.5.4.3.6 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 13-75 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.

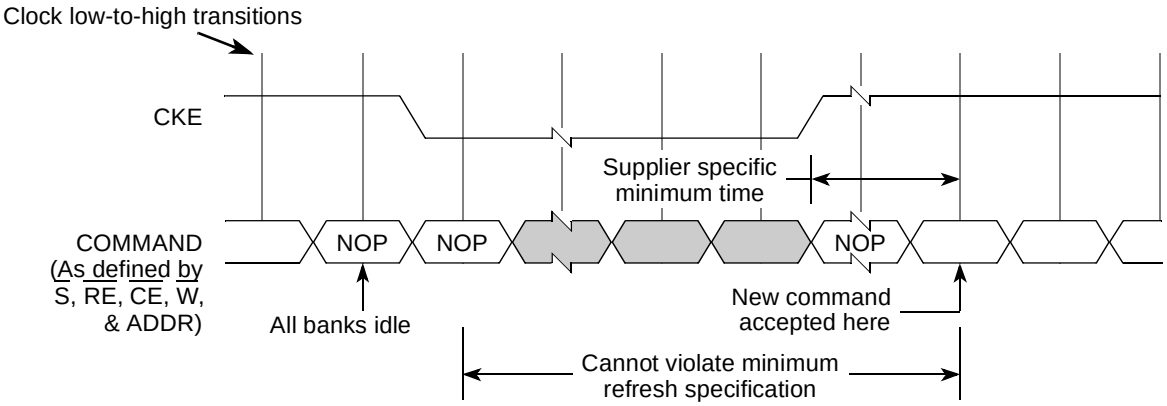


Figure 13-75. SDRAM Power-Down Timing

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

13.5.4.3.7 Self-Refresh

In order to be able to stop activity on the local bus (for power save or debug), while the content of the SDRAM is maintained, the self-refresh mode is supported. This mode is invoked by issuing a

self-refresh command to the SDRAM. The LBC applies the same timing as for the auto refresh, but also pulls 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 13-76 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 13.4.3.3, “Intel PC133 and JEDEC-Standard SDRAM Interface Commands,” for SDRAM interface commands and information on the self-refresh command.

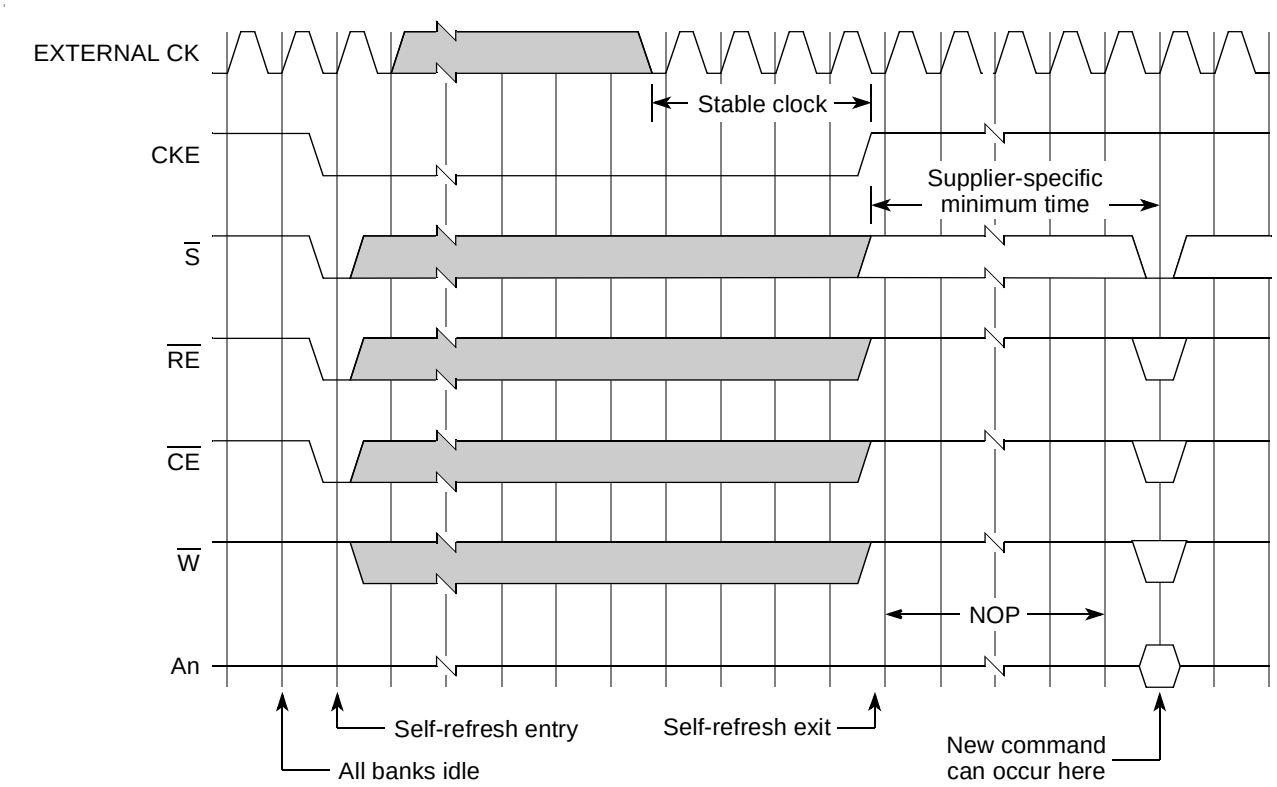


Figure 13-76. SDRAM Self-Refresh Mode Timing

13.5.4.3.8 SDRAM Timing

To allow for very high speeds on the memory bus, the capacitive loading on the local bus must be taken into consideration as shown in Table 13-44.

Table 13-44. 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 13-45. 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 13-77 shows relative timings for the local bus clock DLL.

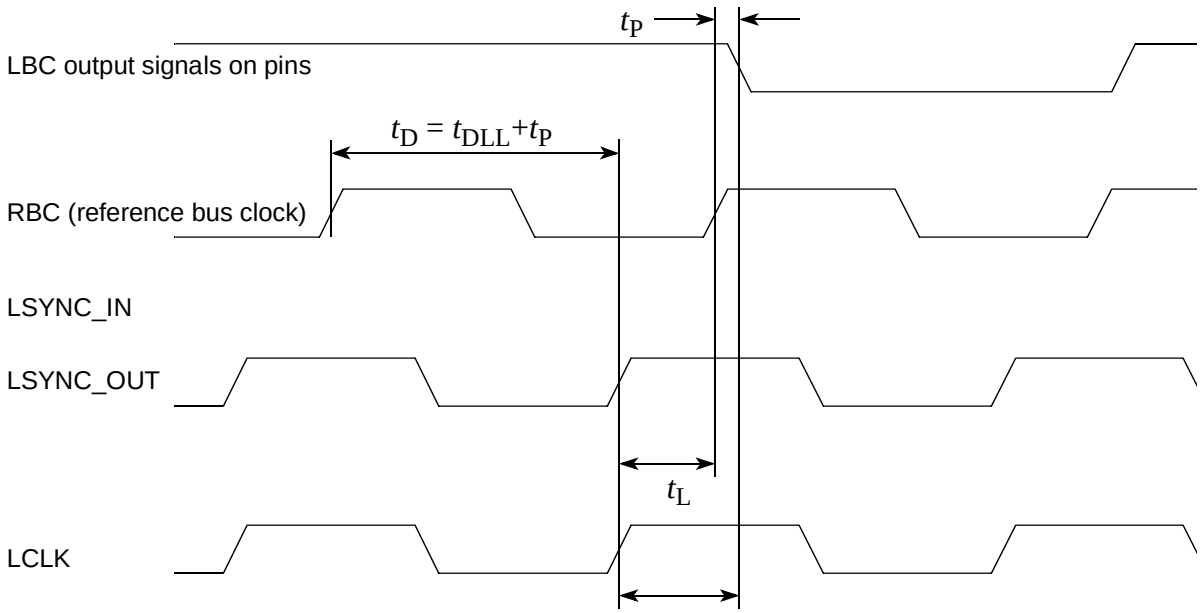


Figure 13-77. Local Bus DLL Operation

13.5.4.4 Parity Support for SDRAM

Contrary to older DRAM technologies, SDRAM devices typically are organized either x4, x8, x16 or x32. There are no mainstream devices that include parity support. To allow for error protection on the local bus an additional SDRAM for the 4 parity bits must be used. Since the local bus allows for SDRAM accesses with less than the full port size, read-modify-write cycles are supported for SDRAM write cycles.

Figure 13-78 shows a connection diagram.

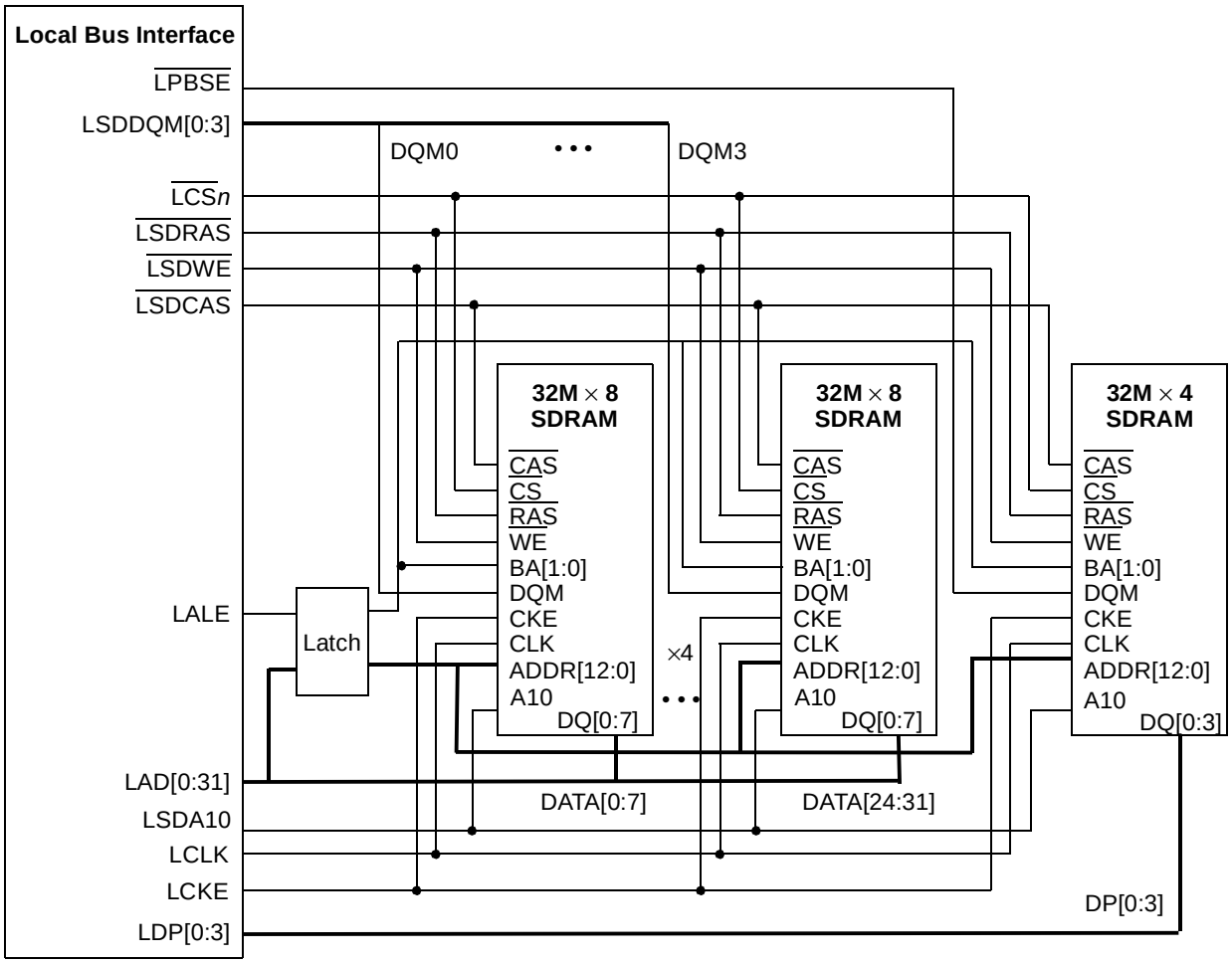


Figure 13-78. Parity Support for SDRAM

13.5.5 Interfacing to ZBT SRAM

In many applications, SDRAM provides sufficient performance for the local bus. However, especially in networking applications, 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 13-79

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.

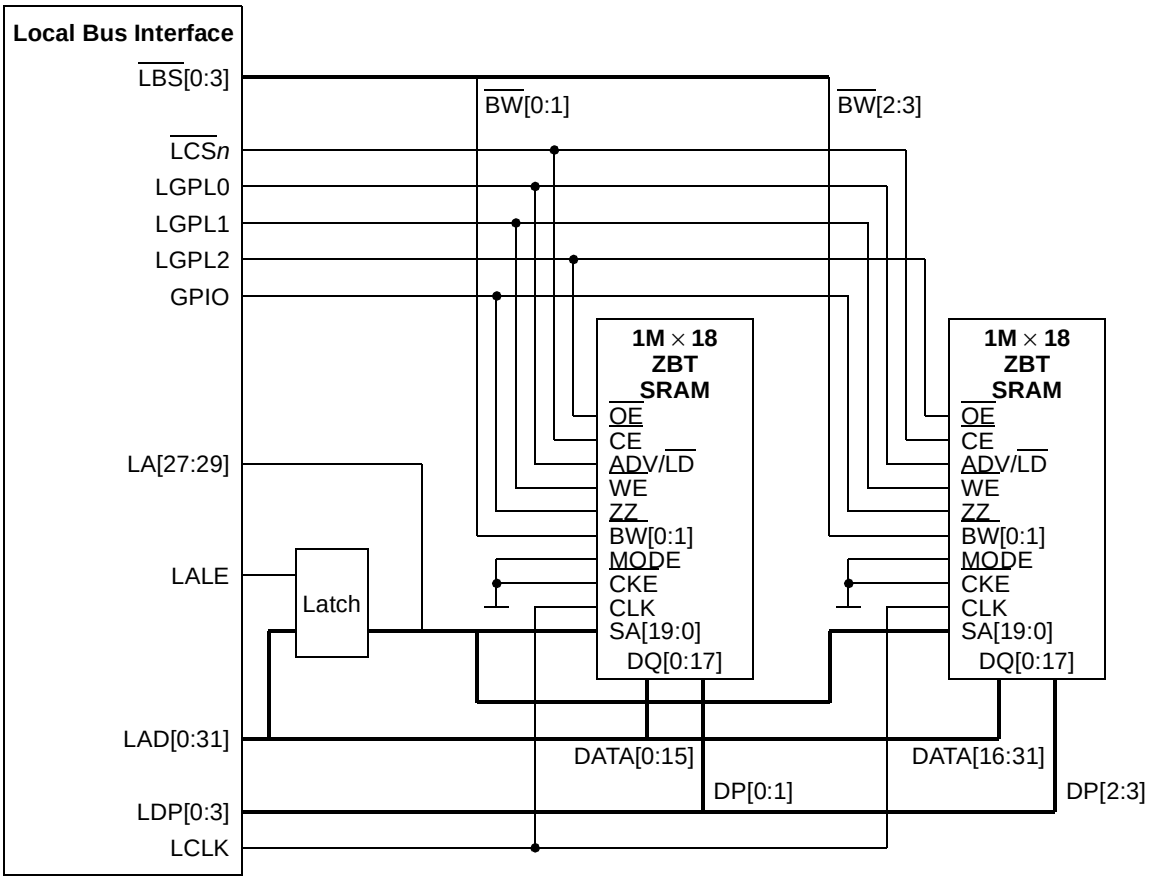


Figure 13-79. Interface to ZBT SRAM

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.

13.5.6 Interfacing to DSP Host Ports

In many applications, an integrated communications processor aggregates traffic for DSPs and distributes that traffic to the DSPs. The local bus allows connection to a variety of different DSP host ports and this section gives some information on how to interface to some example DSPs.

13.5.6.1 Interfacing to MSC8101 HDI16

This section describes how to interface to the HDI16 peripheral interface of the MSC8101. After initial set-up of the 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.

13.5.6.1.1 HDI16 Peripherals

The host interface (HDI16) is a 16-bit-wide, full-duplex, double-buffered parallel port that can directly connect to the data bus of a host processor. 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 13-81 illustrates the relationship between the two sides.

All of the HDI16 peripheral's registers are mapped directly onto the MSC8101's QBus, as defined by the *MSC8101 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 13.5.6.1.2, "Physical Interconnections."

The HDI16 host port itself is a 16-bit-wide parallel port with various strobe and multiplexing options.

The most important HDI16 host port facet is that it is specified as an asynchronous interface and so reduces concerns over clock skew between the HDI16 host port and the host device's buses. Furthermore, with all the host port registers being accessed with a single chip select and 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 13-80 shows the internal register diagram of the HDI16.

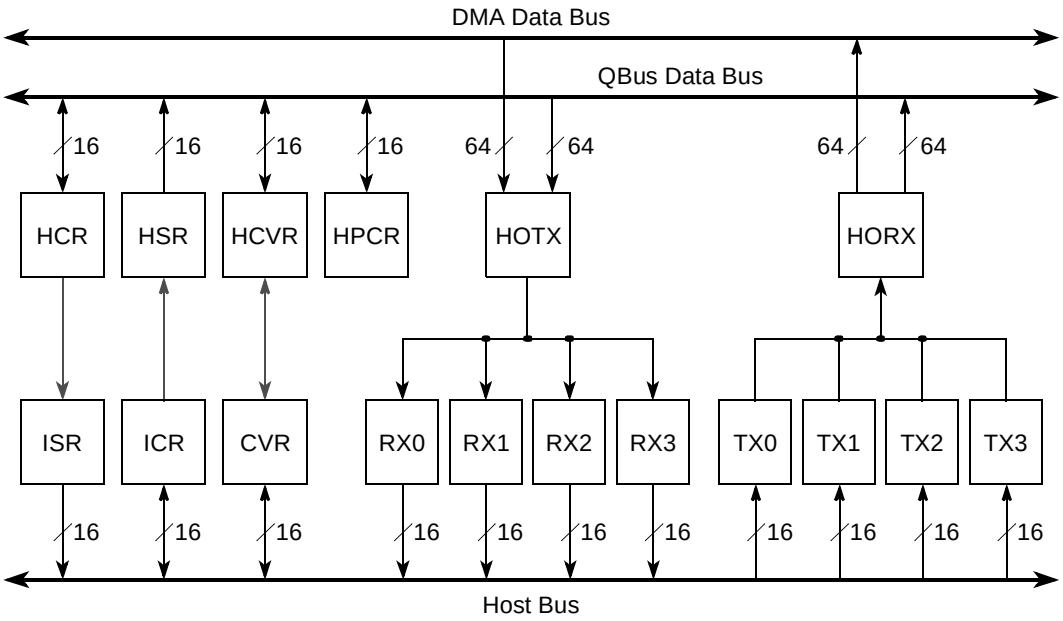


Figure 13-80. MSC8101 HDI16 Peripheral Registers

13.5.6.1.2 Physical Interconnections

The physical interconnections between the UPM controlled local bus and the HDI16 of the MSC8101 HDI16 peripheral are given in Table 13-46 and Figure 13-81.

Table 13-46. 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

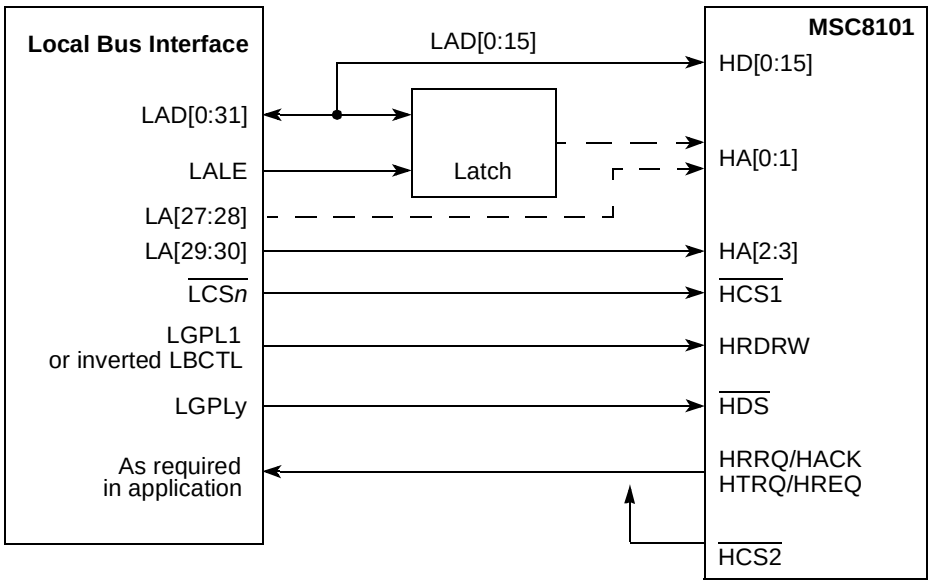


Figure 13-81. Interface to MSC8101 HDI16

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.

13.5.6.1.3 Supporting Burst Transfers

As mentioned previously, to facilitate burst transfers the host's local bus address lines can be connected in a very specific way. First, the local bus A31 signal is not required, as the HDI16 registers are 16-bit word addressed. Secondly, local bus LA[27] and LA[28] signals can be eliminated, so that the host side transmit and receive registers wrap around the same four 16-bit word addresses.

For example, host transmit register 0 on the HDI16 peripheral (address 0x04) can be obtained by the host by accessing any of the following memory mapped addresses: 0x20, 0x28, 0x30, or 0x38. This is critical for burst accesses as the source or destination addresses increment after each 16-bit

access to the interface for all 16 transactions within that burst. By using the addressing as defined, if the first access is at 0x20, the last will be at 0x3e but, more importantly, the four host Tx/Rx registers will have been looped around four times.

13.5.6.1.4 Host 60x Bus: HDI16 Peripheral Interface Hardware Timings

The host UPM–controlled local bus and the HDI16 MSC8101's HDI16 host interface are both programmable. Careful programming of the host chip select registers and UPM can meet the HDI16 MSC8101 host port timings.

On any bus access the critical timing for both read and write is typically around the data latch point. For the UPM based read access, the host has the flexibility to latch data on a rising or falling LCLK edge. The falling LCLK edge is used here to latch the HDI16 data into the host MSC8101 at its earliest convenience.

After the data is latched, appropriate HDI16 port data hold time is ensured before the data strobe (DS) and chip select (CS1) are negated.

On a UPM write cycle, the critical action is in enveloping the DS assertion with CS asserted to ensure proper write data hold time after latching by the HDI16 host port.

Special attention needs to be given to both the host read and write access strobe (DS) negation times (HDS assert).

The HDI16 MSC8101 specifies some restrictions for consecutive register access, which results in a hold off negation time for the read and write access strobes.

Rather than restrict the firmware to avoid consecutive bus accesses to host port registers, the negation hold off times should be accommodated in the UPM hardware interface settings. Additional clocks must be built into the end of UPM based cycle giving appropriate time before the next bus cycle starts.

The timings can be readily adapted to allow external decode logic to be added to support chip selects for a larger number of DSP HDI16 host ports.

13.5.6.2 Interfacing to MSC8102 DSI

The MSC8102 direct-slave interface (DSI) gives an external host direct access to the MSC8102. It provides the following slave interfaces to an external host:

- Asynchronous SRAM-like interface giving the host single accesses (with no external clock).
- Synchronous SSRAM-like interface giving the host single or burst accesses of 256 bits (eight beats of 32 bits or four beats of 64 bits) with its external clock decoupled from the MSC8102 internal bus clock.

The DSI supports 32- or 64-bit data bus. For connection to the local bus the DSI has to be configured in 32-bit mode. This is achieved through the DSP reset configuration.

The DSI supports two addressing modes, which are determined during the MSC8102 boot sequence. Refer to details in the MSC8102 documentation.

- Full address bus mode with HA[11:29] used in both 32-bit data mode and 64-bit data mode
- Sliding window mode with HA[14:29] used in both 32-bit data mode and 64-bit data mode

13.5.6.2.1 DSI in Asynchronous SRAM-Like Mode

The local bus supports the DSI single strobe as well as the DSI double strobes of operation. As an example the dual strobe configuration is shown below.

Figure 13-82 shows the interface to the MSC8102 DSI for asynchronous mode.

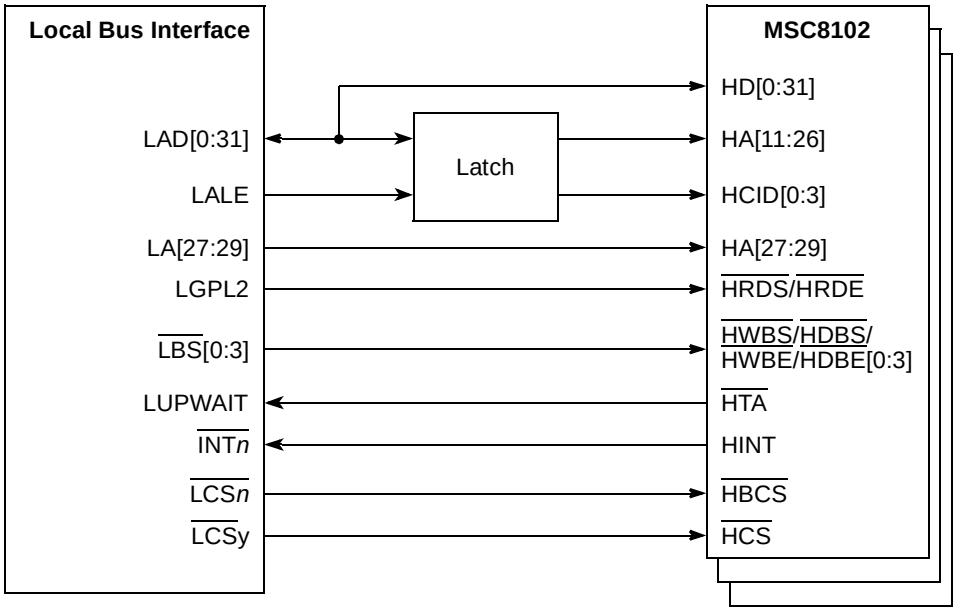


Figure 13-82. Interface to MSC8102 DSI in Asynchronous Mode

The asynchronous SRAM-like mode of the DSI is inherently slower than the synchronous mode and should be used, if only relatively small amounts of data are transferred between the communications controller and the MSC8102. To allow for maximum timing flexibility, the UPM machine of the 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 13-83 shows an asynchronous write access. The DSI samples the host chip ID signals (HCID[0:3]) on the first falling edge of the host write byte strobe signals ($\overline{\text{HWBS}}$) on which the host chip select signal ($\overline{\text{HCS}}$) is asserted. If the HCID[0:3] signals match the CHIPID value, the

DSI is accessed. The DSI will signal with the assertion of the host transfer acknowledge signal ($\overline{\text{HTA}}$), whether it is ready to sample the host data bus (HD), and the host can terminate the access by immediately negating $\overline{\text{HWBS}}$. The WAEN feature of the UPM must be used to insert wait states while the DSI is busy. The UWPL bit in the 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.

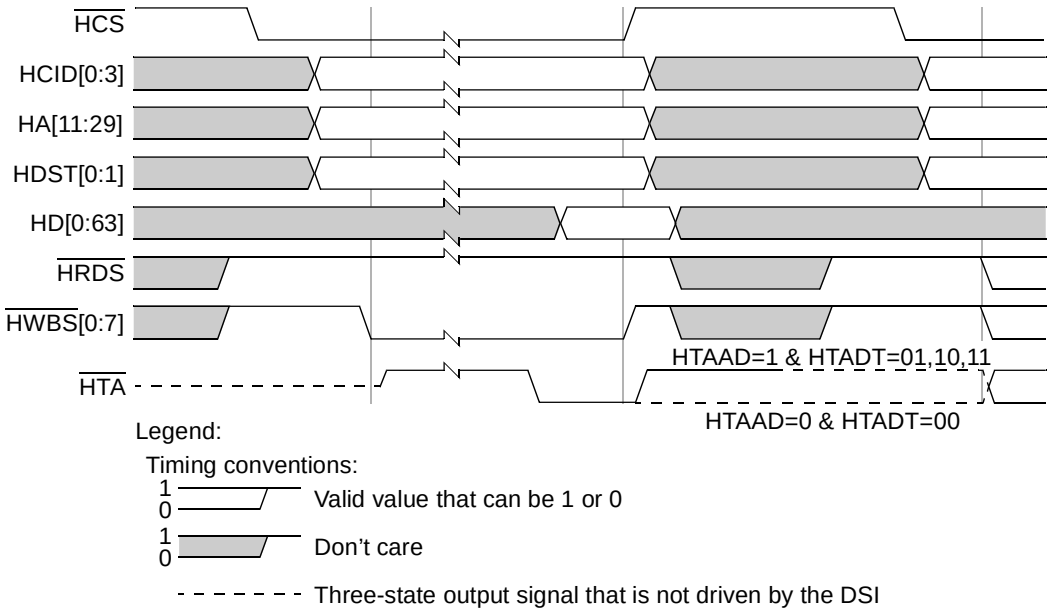


Figure 13-83. Asynchronous Write to MSC8102 DSI

Figure 13-84 shows an asynchronous read access. The DSI samples the host address bus (HA) and the HCID on the first falling edge of the host read strobe signal ($\overline{\text{HRDS}}$) on which the $\overline{\text{HCS}}$ is asserted. If HCID[0:3] match the CHIPID value, the DSI is accessed. When the DCR[8]:RPE bit is set, read access to the memory space (not to the register space) initiates data prefetching from consecutive addresses in the internal memory space. The DSI signals (with the assertion of the host transfer acknowledge signal ($\overline{\text{HTA}}$)) that data is valid, and the host can sample the host data bus

(HD) and terminate the access by negating 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.

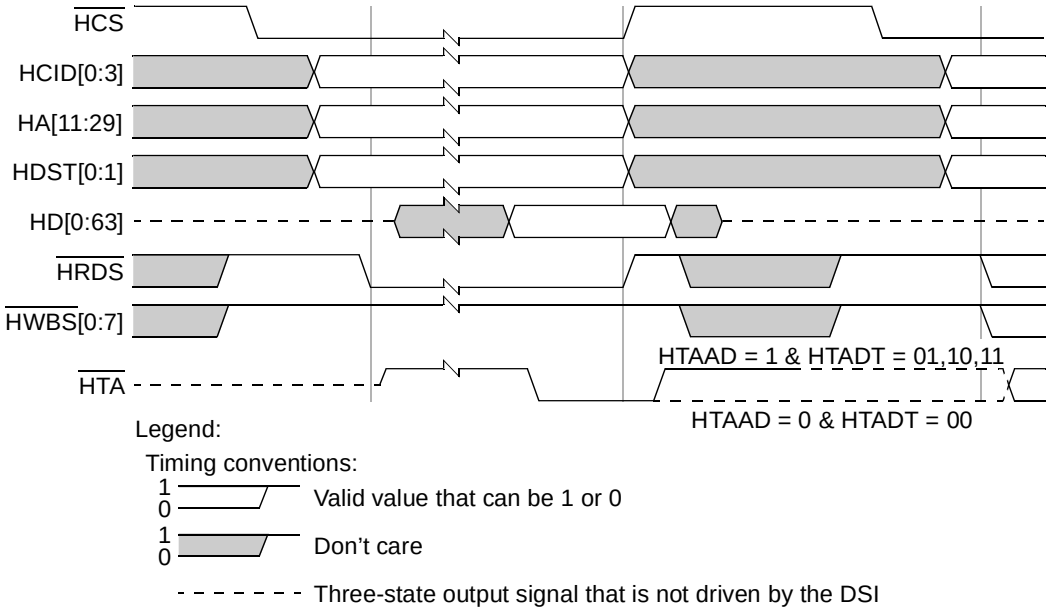


Figure 13-84. Asynchronous Read from MSC8102 DSI

13.5.6.2.2 DSI in Synchronous Mode

The synchronous SSRAM-like mode of the DSI is inherently faster than the asynchronous mode and should be used if larger amounts of data are transferred between the communications controller and the MSC8102. This will optimize the bus utilization, especially if several MSC8102s are connected to one local bus. The UPM machine of the LBC must be used to implement this interface.

Figure 13-85 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 13.4.4.4.10, "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.

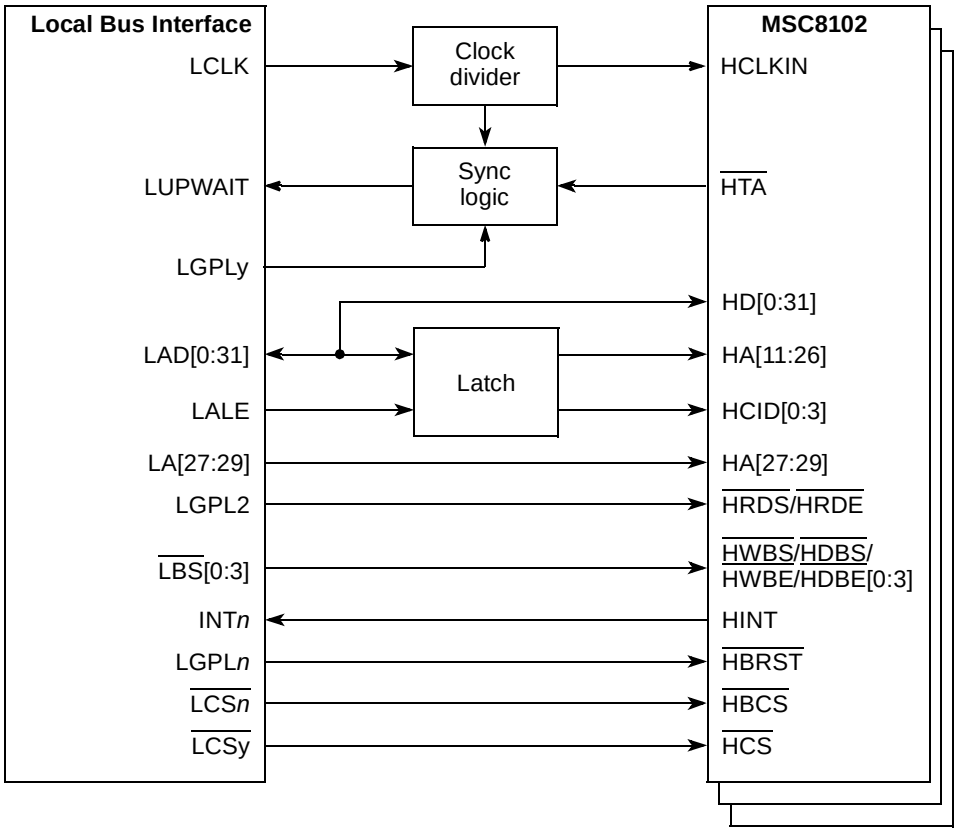


Figure 13-85. Interface to MSC8102 DSI in Synchronous Mode

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 13.4.4.5, "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 13-86 shows an example for a synchronization mechanism for a clock divider of 3. Note that the length of the synchronization cycle depends on the relative start of the synchronization process and varies with every access. It can vary in length from one to n (clock ratio) local bus clocks.

The second column (compensation cycle) is intended to compensate for the reaction time of 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.

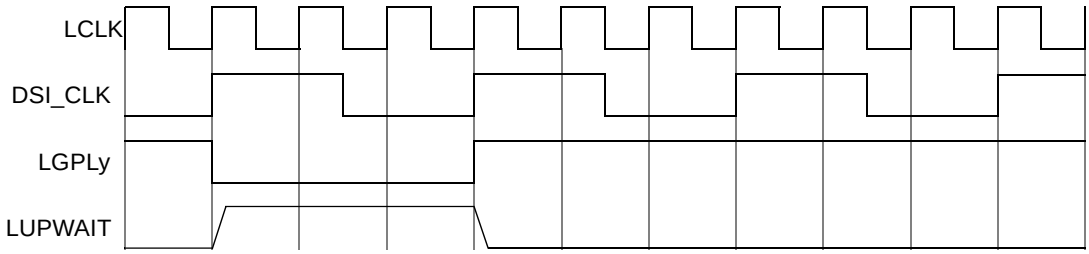


Figure 13-86. UPM Synchronization Cycle

Table 13-47. 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 13-47. 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 13-87 shows a synchronous single write access.

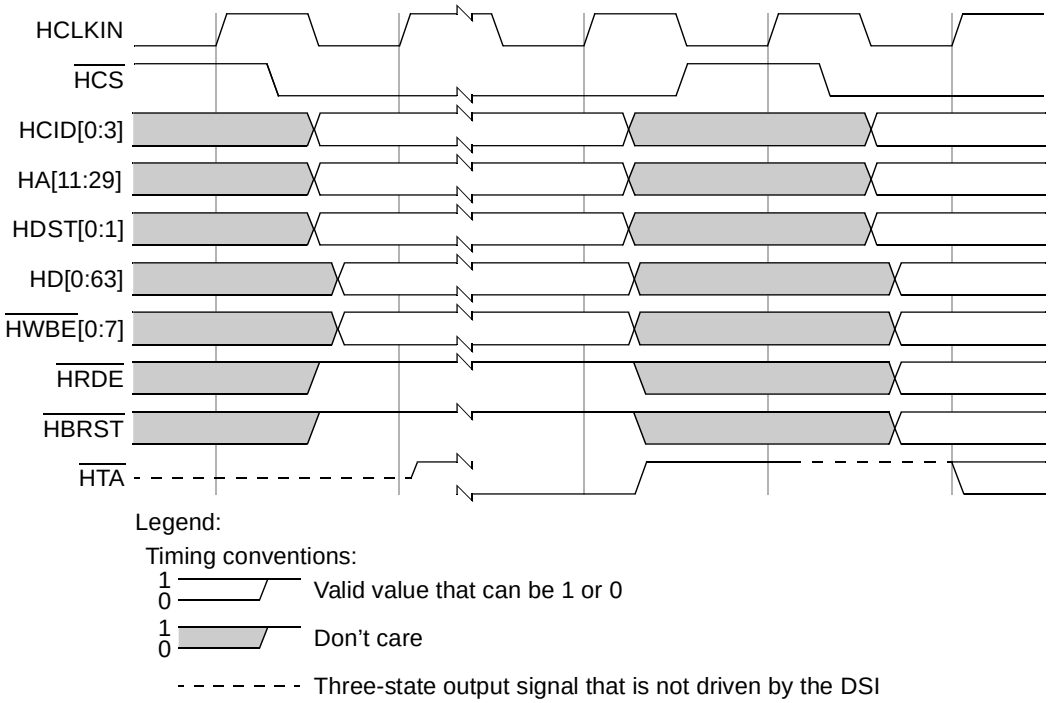


Figure 13-87. Synchronous Single Write to MSC8102 DSI

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 13-88 shows a synchronous single read access.

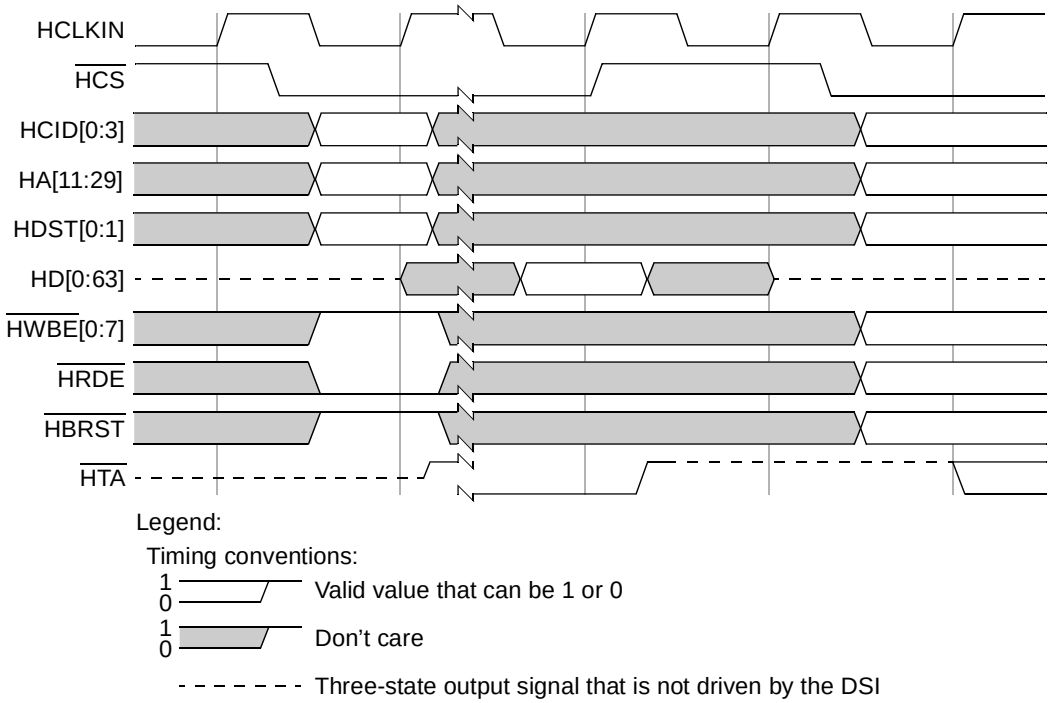


Figure 13-88. Synchronous Single Read from MSC8102 DSI

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 13-89 shows a synchronous burst write access.

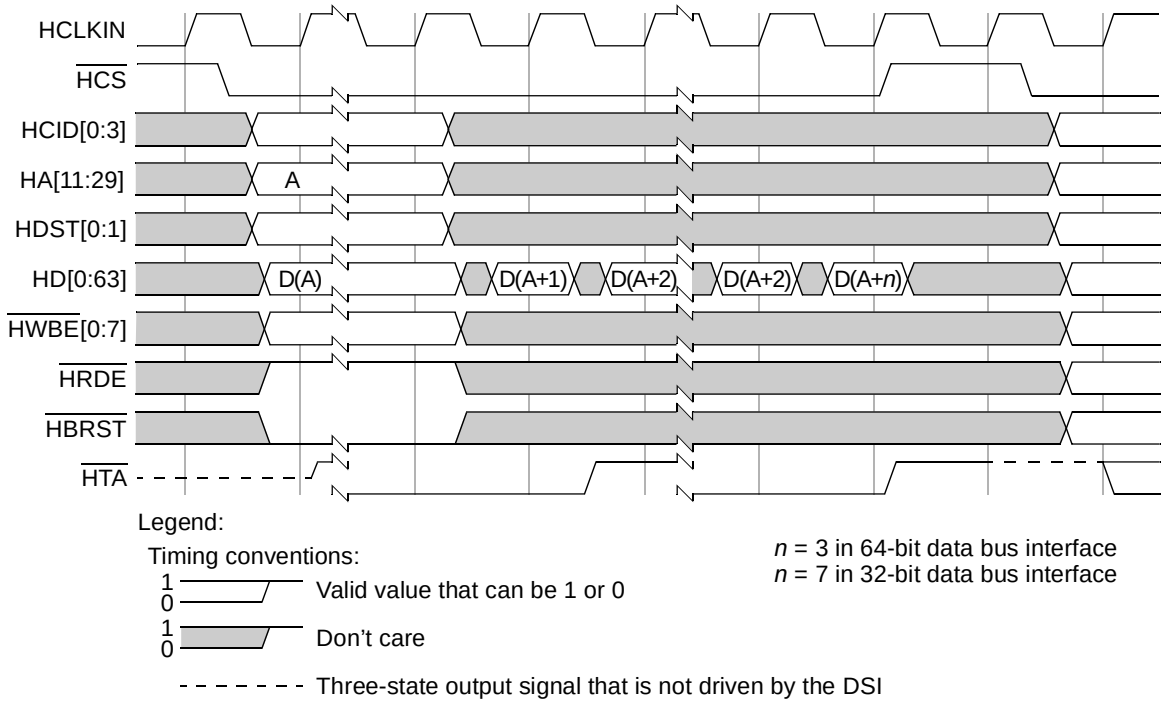


Figure 13-89. Synchronous Burst Write to MSC8102 DSI

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. HWBE are asserted, HBRST is asserted, and HRDE is negated. Assertion of HTA indicates that the DSI is ready to complete the current beat of the access and the host must proceed to the next beat of this access. When the host reaches the last beat of the access, it must terminate the burst access. Typically HTA is asserted immediately for each beat of the access. If the write buffer is full, HTA assertion is delayed. Because HTA is connected to the LUPWAIT signal of the UPM, all local bus signals are frozen until HTA goes to 0 and then the UPM continues in its pattern. After the last beat of the access, HTA is driven to logic 1 and stops being driven on the next rising edge of HCLKIN. The host can start its next access to the same MSC8102 immediately in the next HCLKIN rising edge without negating HCS between accesses. If the next access is not to the same MSC8102, then, to prevent contention on HTA, the host must wait to access the next device until the previous DSI stops driving HTA. The easiest way to achieve this is to insert idle cycles at the end of the UPM pattern to guarantee that HTA is inactive.

Synchronous Burst Read

Figure 13-90 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.

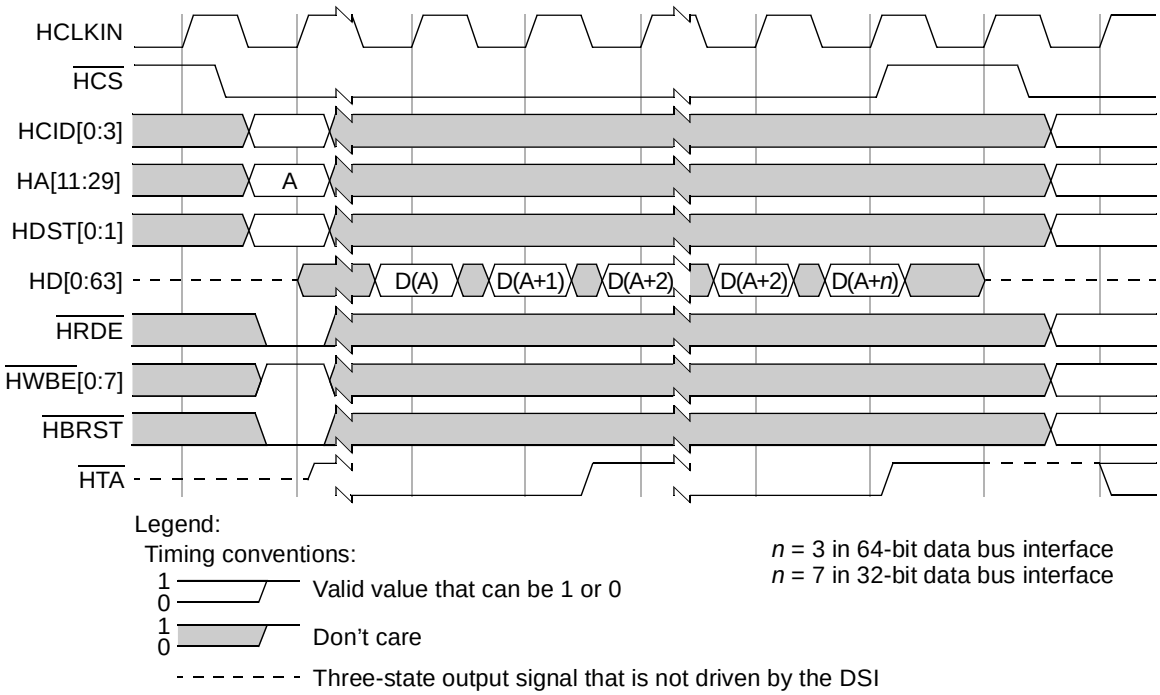


Figure 13-90. Synchronous Burst Read from MSC8102 DSI

$\overline{\text{HRDE}}$ 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.

13.5.6.2.3 Broadcast Accesses

Using $\overline{\text{HBCS}}$, a host can share one chip-select signal between multiple MSC8102 devices for broadcasting write accesses. In broadcast mode, the DSI does not drive its $\overline{\text{HTA}}$ signal to prevent contention between multiple devices driving different values to the same signal. Also, the DSI does not decode 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.

13.5.6.3 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 13-48.

Table 13-48. 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.

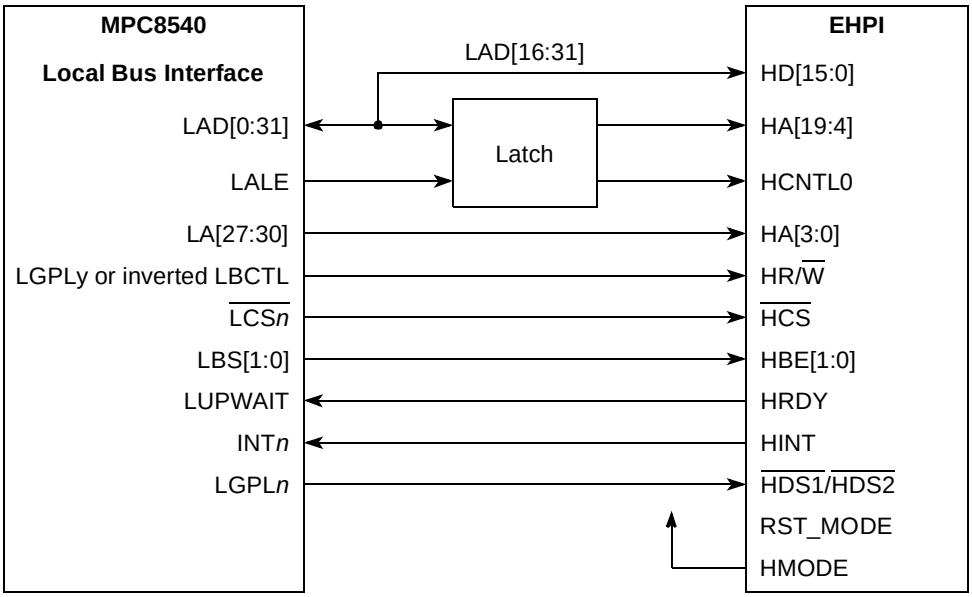


Figure 13-91. Interface to Texas Instruments EHPI in Non-Multiplexed Mode

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 13-92 shows read timing required by the EHPI in non-multiplexed mode.

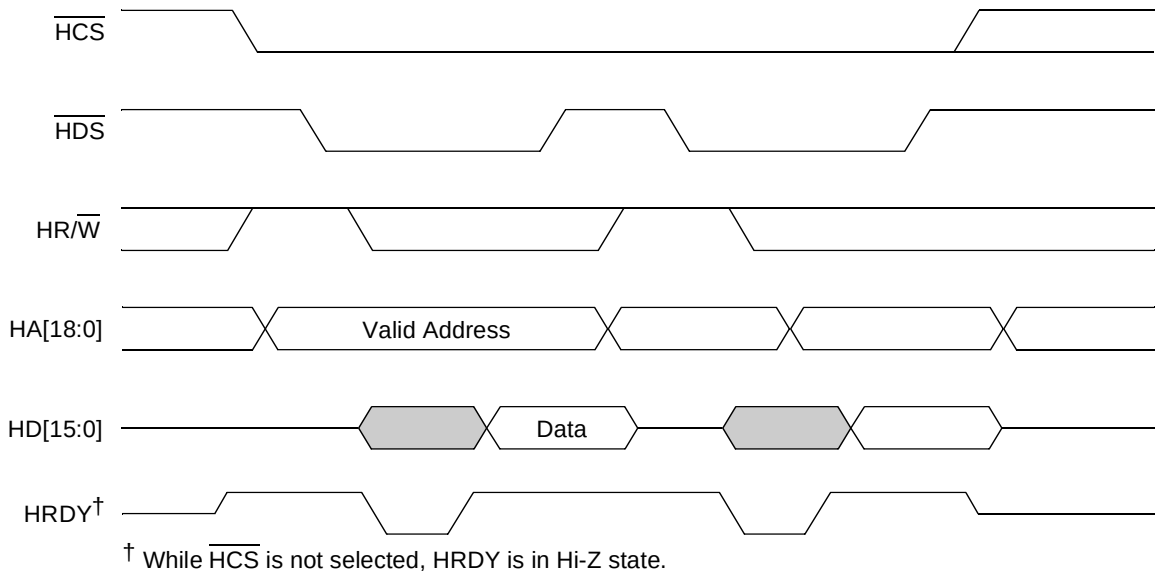


Figure 13-92. EHPI Non-Multiplexed Read Timings

Figure 13-93 shows write timing required by the EHPI in non-multiplexed mode.

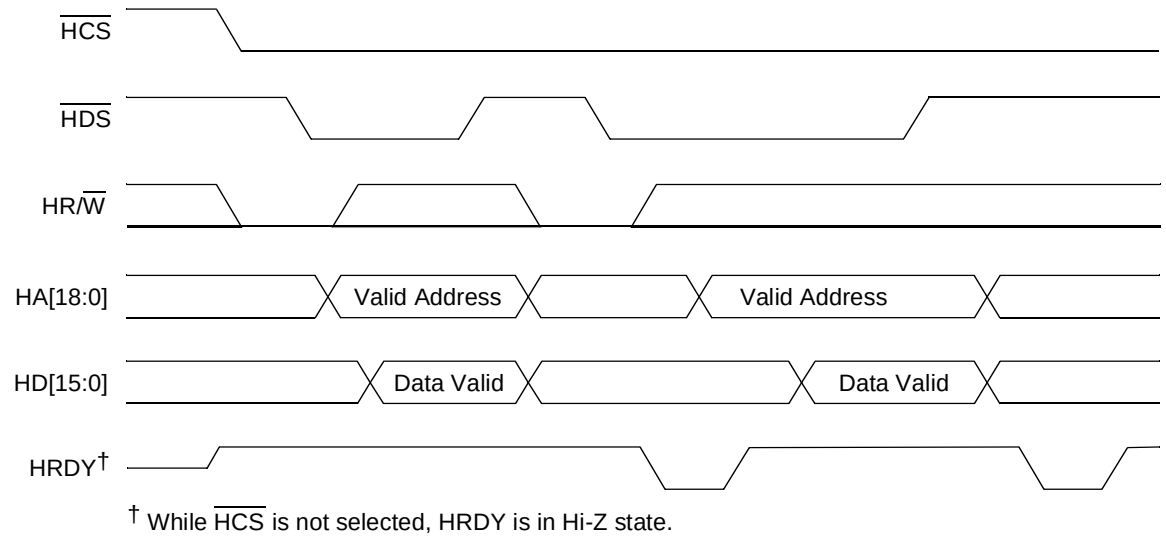


Figure 13-93. EHPI Non-Multiplexed Write Timings

13.5.6.3.1 Expansion to Multiple DSPs

The connection shown above can be adapted easily to interface to multiple DSPs instead of only one. Each DSP host port needs to receive its own chip select and interrupt signals, and HRDY must be connected differently, depending on which DSP is used. All other signals can be connected to all host ports in parallel.

HRDY signals can be bussed 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 14
Three-Speed Ethernet Controllers

This chapter describes the two three-speed Ethernet controllers of the MPC8540. The two controllers are referenced as TSEC1 and TSEC2. Note that this chapter does not include the description of the separate 10/100 Ethernet controller (FEC) on the MPC8540.

14.1 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 14-1 shows the typical Ethernet protocol stack and the relationship to the OSI model.

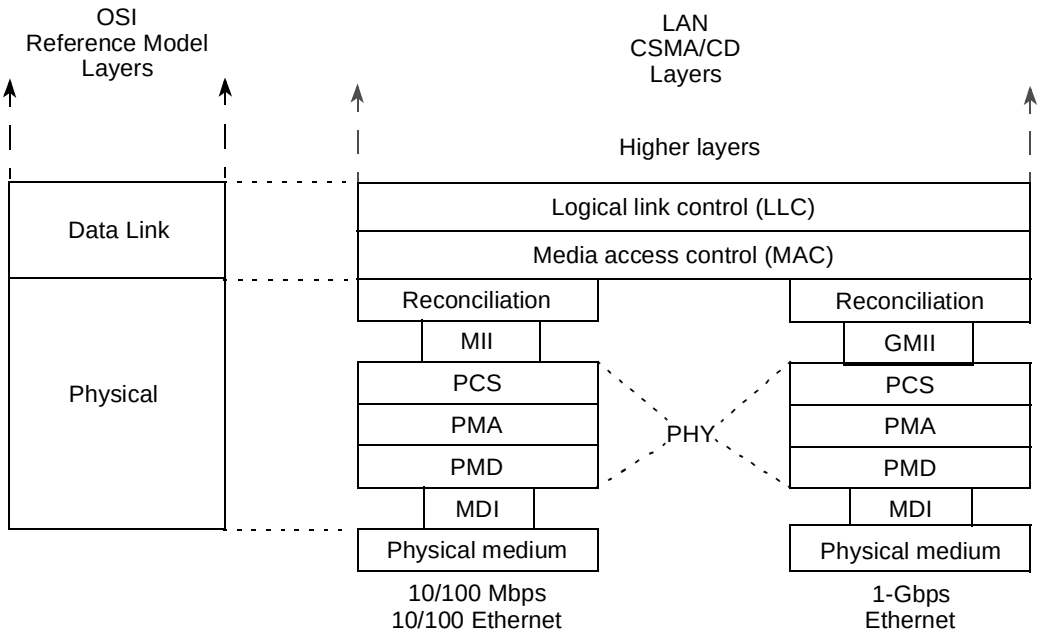


Figure 14-1. Ethernet Protocol in Relation to the OSI Protocol Stack

Gigabit Ethernet provides the following sublayers:

- Media access control (MAC) sublayer—The MAC sublayer provides a logical connection between the MAC and its peer station. Its primary responsibility is to initialize, control, and manage the connection with the peer station.
- Reconciliation sublayer—The reconciliation sublayer acts as a command translator. It maps the terminology and commands used in the MAC layer into electrical formats appropriate for the physical layer entities.
- MII (media-independent interface) sublayer—The MII sublayer provides a standard interface between the MAC layer and the physical layer for 10/100 Mbps operations. It isolates the MAC layer and the physical layer, enabling the MAC layer to be used with various implementations of the physical layer.
- GMII (gigabit media-independent interface) sublayer—The GMII sublayer provides a standard interface between the MAC layer and the physical layer for 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 14-2 describes the different physical interface standards.

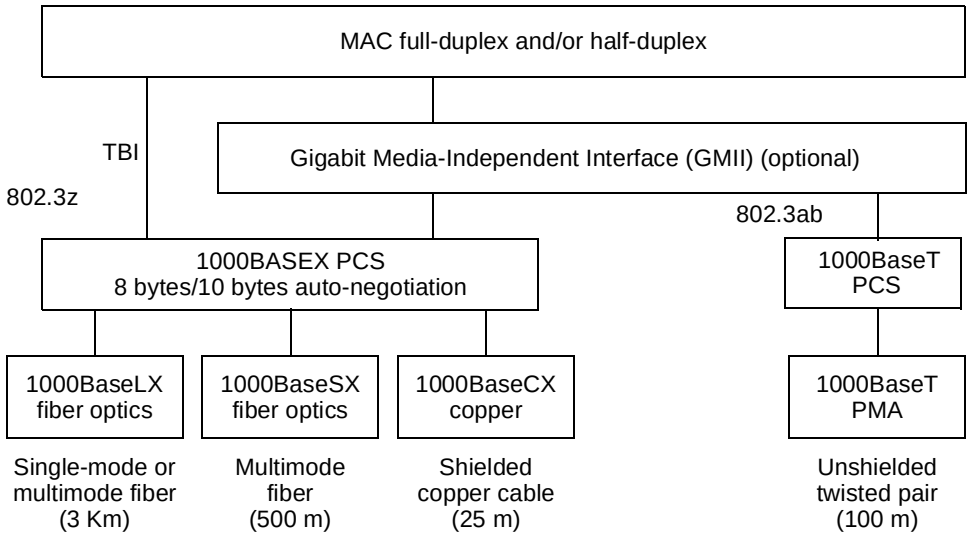


Figure 14-2. IEEE 802.3z and 802.3ab Physical Standards

Ethernet/IEEE 802.3 frames are based on the frame structure shown in Figure 14-3. The term 'packet' is sometimes used to refer to the frame plus the preamble and start frame delimiter (SFD).

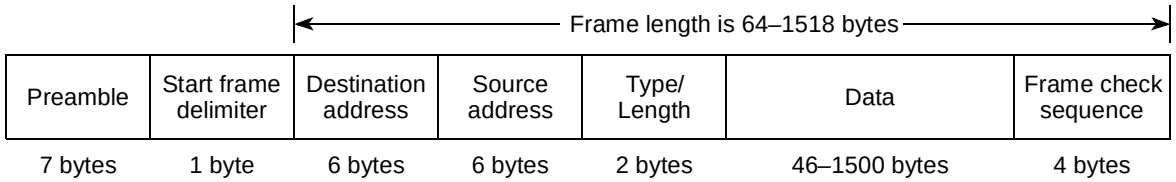


Figure 14-3. Ethernet/IEEE 802.3 Frame Structure

The elements of an Ethernet frame are as follows:

- Preamble— The 7-byte preamble of alternating ones and zeros used for receiver timing synchronization (each byte containing the value 0x55).
- Start frame delimiter (SFD)—A sequence of 0xD5 (10101011 because the bit ordering is lsb first) indicates the beginning of the frame.
- 48-bit destination address (DA)—The first bit identifies the address as an individual address (0) or a group address (1). The second bit is used to indicate whether the address is locally-defined (1) or globally-defined (0).
- 48-bit source address (SA)— (Original versions of the IEEE 802.3 specification allowed 16-bit addressing, which has never been used widely.)
- Ethernet type field/IEEE 802.3 length field—The type field signifies the protocol (for example, TCP/IP) used in the rest of the frame. The length field specifies the length of the data portion of the frame. For both Ethernet and IEEE 802.3 frames to exist on the same LAN, the length field must be unique from any type fields used in Ethernet. This limitation requires that a type field be identified by a decimal number equal to or greater than 1536 (0x0600) but less than 65535 (0xFFFF). If the number, however, is between 0 and 1,500 (0x0000 through 0x05DC) then this field indicates the length of the MAC client data. The range from 1,501 to 1,535 (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 14-4 provides additional details of the Ethernet/IEEE 802.3 frame structure.

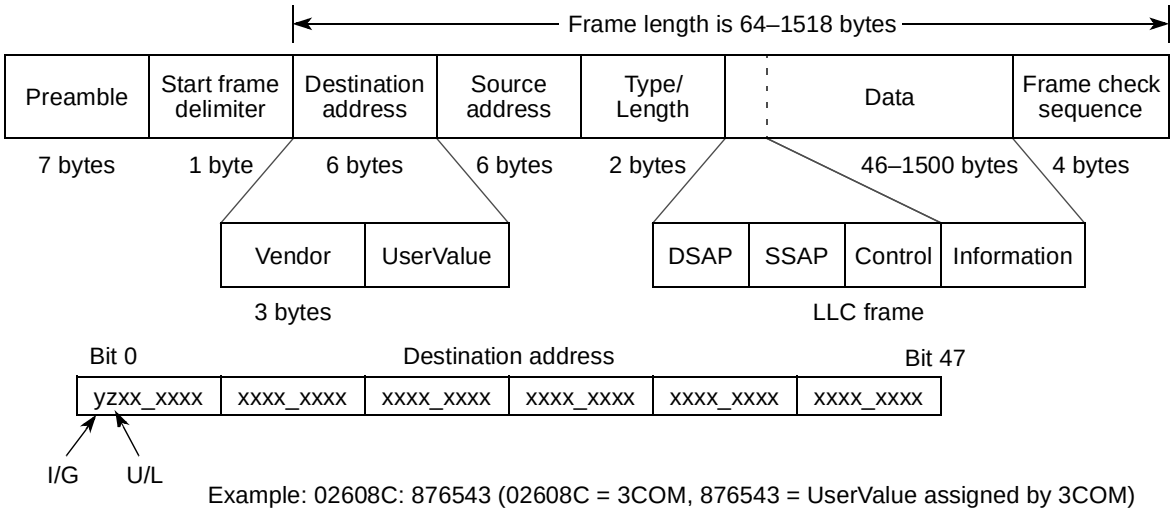


Figure 14-4. Ethernet/IEEE 802.3 Frame Structure With More Details

Relative to Figure 14-4, the IEEE 802.3 section 3.11 (MAC frame format) defines the frame format such that the octets of a frame are transmitted from left to right (preamble first, the FCS last), and the bits of each octet are transmitted least-significant bit (lsb) first. The destination address example shown in Figure 14-4 (02608C:876543) which would normally be written as

0000 0010 0110 0000 1000 1100 1000 0111 0110 0101 0100 0011

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 14-5 shows the block diagram of the TSEC.

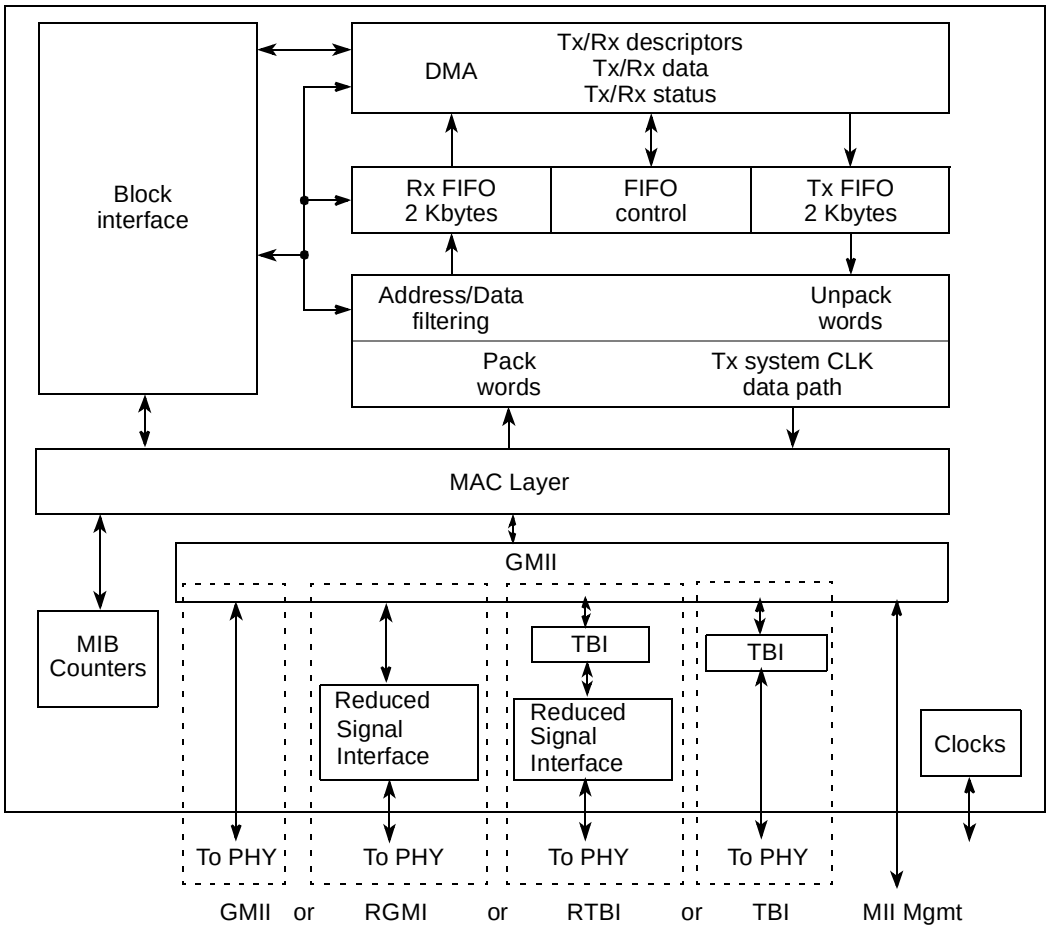


Figure 14-5. TSEC Block Diagram

14.1.1 Three-Speed Ethernet Controller Overview

The TSEC is designed to support 10, 100, and 1000 Mbps Ethernet/802.3 networks and contains the following components:

- Ethernet media access controller (MAC)
- First-in first-out (FIFO) controller
- Direct memory access (DMA) controller
- Ten-bit interface (TBI)
- Register-based statistical module that supports management information base (MIB) remote monitoring (RMON)

The most-significant byte of data in a receive or transmit data buffer corresponds to the most-significant byte of a frame, respectively.

The complete TSEC is designed for single MAC applications. The TSEC supports several standard MAC-PHY interfaces to connect to an external Ethernet transceiver:

- MII interface running at 10/100 Mbps
- GMII interface running at 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.

14.2 Features

The MPC8540 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

14.3 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 14.5.3.8.1, “Individual Address Registers 0–7 (IADDRn),” Section 14.5.3.8.2, “Group Address Registers 0–7 (GADDRn),” Section 14.5.3.4.1, “Receive Control Register (RCTRL),” and Section 14.6.2.6, “Frame Recognition.”

- RMON support

  See Section 14.6.2.5, “RMON Support.”

- Internal loop back

  Internal loop back mode is selected through the Loop Back bit in the MACCFG1 register. See Section 14.6.2.10, “Internal and External Loop Back,” for details.

14.4 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.

14.4.1 Detailed Signal Descriptions

Table 14-1 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 14-1. 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 14-1. 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 14-1. 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 MPC8540. 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.

14.5 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 14.6.3, "Buffer Descriptors."

The ten-bit interface (TBI) module MII registers are also described in this section. The TBI registers are defined like PHY registers and, as such, are accessed via the MII management interface in the same way as the PHYs are accessed. For detailed descriptions of the TBI registers (the MII register set for the ten-bit interface) please refer to Section 14.5.4, "Ten-Bit Interface (TBI)."

14.5.1 Top-Level Module Memory Map

The TSEC implementation requires 4 Kbytes of memory-mapped space, of which more than 1 Kbyte is reserved for future expansion. The space is divided into the following sections:

- General control/status registers
- Transmit-specific control/status registers
- Receive-specific control/status registers
- MAC registers
- Event/statistic counters held in the MIB block
- Hash function registers

Table 14-2 defines the top-level memory map.

Table 14-2. Module Memory Map Summary

Address	Function
000–0FF	TSEC general control/status registers
100–2FF	TSEC transmit control/status registers
300–4FF	TSEC receive control/status registers
500–5FF	TSEC MAC registers

Table 14-2. 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

14.5.2 Detailed Memory Map—Control/Status Registers

Table 14-3 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 14-3, but the registers for TSEC2 are not. Note in Table 14-3 that the registers are the same for TSEC2 except that the offset changes from 0x2_4*nnn* to 0x2_5*nnn*.

Table 14-3. Module Memory Map

Offset	Name	Access[1]	Reset	Section/Page
TSEC1 General Control and Status Registers				
0x2_4000–0x2_400C	Reserved	R	0x0000_0000	—
0x2_4010	IEVENT—Interrupt event register	R/W	0x0000_0000	14.5.3.1.1/14-20
0x2_4014	IMASK—Interrupt mask register	R/W	0x0000_0000	14.5.3.1.2/14-23
0x2_4018	EDIS—Error disabled register	R/W	0x0000_0000	14.5.3.1.3/14-24
0x2_401C	Reserved	R	0x0000_0000	—
0x2_4020	ECNTRL—Ethernet control register	R/W	0x0000_0000	14.5.3.1.4/14-25
0x2_4024	MINFLR—Minimum frame length register	R/W	0x0000_0040	14.5.3.1.5/14-26
0x2_4028	PTV—Pause time value register	R/W	0x0000_0000	14.5.3.1.6/14-27
0x2_402C	DMACTRL—DMA control register	R/W	0x0000_0000	14.5.3.1.7/14-28
0x2_4030	TBIPA—TBI PHY address register	R/W	0x0000_0000	14.5.3.1.8/14-29
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	14.5.3.2.1/14-30
0x2_4050–0x2_4088	Reserved	R	0x0000_0000	—
0x2_408C	FIFO_TX_THR—FIFO transmit threshold register	R/W	0x0000_0100	14.5.3.2.2/14-31

Table 14-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_4090–0x2_4094	Reserved	R	0x0000_0000	—
0x2_4098	FIFO_TX_STARVE—FIFO transmit starve register	R/W	0x0000_0080	14.5.3.2.3/14-32
0x2_409C	FIFO_TX_STARVE_SHUTOFF—FIFO transmit starve shutoff register	R/W	0x0000_0100	14.5.3.2.4/14-32
0x2_40A0–0x2_40FC	Reserved	R	0x0000_0000	—
TSEC1 Transmit Control and Status Registers				
0x2_4100	TCTRL—Transmit control register	R/W	0x0000_0000	14.5.3.3.1/14-33
0x2_4104	TSTAT—Transmit status register	R/W	0x0000_0000	14.5.3.3.2/14-34
0x2_4108	Reserved	R	0x0000_0000	—
0x2_410C	TBDLEN—TxBD data length register	R	0x0000_0000	14.5.3.3.3/14-35
0x2_4110	TXIC—Transmit interrupt coalescing configuration register	R/W	0x0000_0000	14.5.3.3.4/14-35
0x2_4114–0x2_4120	Reserved	R	0x0000_0000	—
0x2_4124	CTBPTR—Current TxBD pointer register	R	0x0000_0000	14.5.3.3.5/14-36
0x2_4128–0x2_4180	Reserved	R	0x0000_0000	—
0x2_4184	TBPTR—TxBD pointer register	R/W	0x0000_0000	14.5.3.3.6/14-37
0x2_4188–0x2_4200	Reserved	R	0x0000_0000	—
0x2_4204	TBASE—TxBD base address register	R/W	0x0000_0000	14.5.3.3.7/14-37
0x2_4208–0x2_42AC	Reserved	R	0x0000_0000	—
0x2_42B0	OSTBD—Out-of-sequence TxBD register	R/W	0x0800_0000	14.5.3.3.8/14-38
0x2_42B4	OSTBDP—Out-of-sequence Tx data buffer pointer register	R/W	0x0000_0000	14.5.3.3.9/14-40
0x2_42B8–0x2_42FC	Reserved	R	0x0000_0000	—
TSEC1 Receive Control and Status Registers				
0x2_4300	RCTRL—Receive control register	R/W	0x0000_0000	14.5.3.4.1/14-41
0x2_4304	RSTAT—Receive status register	R/W	0x0000_0000	14.5.3.4.2/14-41
0x2_4308	Reserved	R	0x0000_0000	—
0x2_430C	RBDLEN—RxBD data length register	R	0x0000_0000	14.5.3.4.3/14-42

Table 14-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_4310	RXIC—Receive interrupt coalescing configuration register	R/W	0x0000_0000	14.5.3.4.4/14-43
0x2_4314–0x2_4320	Reserved	R	0x0000_0000	—
0x2_4324	CRBPTR—Current RxBD pointer register	R	0x0000_0000	14.5.3.4.5/14-44
0x2_4328–0x2_433C	Reserved	R	0x0000_0000	—
0x2_4340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	14.5.3.4.6/14-44
0x2_4344–0x2_4380	Reserved	R	0x0000_0000	—
0x2_4384	RBPTR—RxBD pointer register	R/W	0x0000_0000	14.5.3.4.7/14-45
0x2_4388–0x2_4400	Reserved	R	0x0000_0000	—
0x2_4404	RBASE—RxBD base address register	R/W	0x0000_0000	14.5.3.4.8/14-45
0x2_4408–0x2_44FC	Reserved	R	0x0000_0000	—
TSEC1 MAC Registers				
0x2_4500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	14.5.3.6.1/14-49
0x2_4504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	14.5.3.6.2/14-51
0x2_4508	IPGIFG—Inter-packet gap/inter-frame gap register	R/W	0x4060_5060	14.5.3.6.3/14-52
0x2_450C	HAFDUP—Half-duplex register	R/W	0x00A1_F037	14.5.3.6.4/14-53
0x2_4510	MAXFRM—Maximum frame length register	R/W	0x0000_0600	14.5.3.6.5/14-54
0x2_4514–0x2_451C	Reserved	R	0x0000_0000	—
0x2_4520	MIIMCFG—MII management configuration register	R/W	0x0000_0000	14.5.3.6.6/14-54
0x2_4524	MIIMCOM—MII Management command register	R/W	0x0000_0000	14.5.3.6.7/14-55
0x2_4528	MIIMADD—MII Management address register	R/W	0x0000_0000	14.5.3.6.8/14-56
0x2_452C	MIIMCON—MII Management control register	W	0x0000_0000	14.5.3.6.9/14-57
0x2_4530	MIIMSTAT—MII Management status register	R	0x0000_0000	14.5.3.6.10/14-57
0x2_4534	MIIMIND—MII Management indicator register	R	0x0000_0000	14.5.3.6.11/14-58
0x2_4538	Reserved	R	0x0000_0000	—
0x2_453C	IFSTAT—Interface status register	R/W	0x0000_0000	14.5.3.6.12/14-58
0x2_4540	MACSTNADDR1—Station address register, part 1	R/W	0x0000_0000	14.5.3.6.13/14-59

Table 14-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_4544	MACSTNADDR2—Station address register, part 2	R/W	0x0000_0000	14.5.3.6.14/14-60
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	14.5.3.7.1/14-61
0x2_4684	TR127—Transmit and receive 65- to 127-byte frame counter register	R/W	0x0000_0000	14.5.3.7.2/14-61
0x2_4688	TR255—Transmit and receive 128- to 255-byte frame counter register	R/W	0x0000_0000	14.5.3.7.3/14-62
0x2_468C	TR511—Transmit and receive 256- to 511-byte frame counter register	R/W	0x0000_0000	14.5.3.7.4/14-62
0x2_4690	TR1K—Transmit and receive 512- to 1023-byte frame counter register	R/W	0x0000_0000	14.5.3.7.5/14-63
0x2_4694	TRMAX—Transmit and receive 1024- to 1518-byte frame counter register	R/W	0x0000_0000	14.5.3.7.6/14-63
0x2_4698	TRMGV—Transmit and receive 1519- to 1522-byte good VLAN frame count register	R/W	0x0000_0000	14.5.3.7.7/14-64
TSEC1 Receive Counters				
0x2_469C	RBYT—Receive byte counter register	R/W	0x0000_0000	14.5.3.7.8/14-64
0x2_46A0	RPKT—Receive packet counter register	R/W	0x0000_0000	14.5.3.7.9/14-65
0x2_46A4	RFCS—Receive FCS error counter register	R/W	0x0000_0000	14.5.3.7.10/14-65
0x2_46A8	RMCA—Receive multicast packet counter register	R/W	0x0000_0000	14.5.3.7.11/14-66
0x2_46AC	RBCA—Receive broadcast packet counter register	R/W	0x0000_0000	14.5.3.7.12/14-66
0x2_46B0	RXCF—Receive control frame packet counter register	R/W	0x0000_0000	14.5.3.7.13/14-67
0x2_46B4	RXPF—Receive PAUSE frame packet counter register	R/W	0x0000_0000	14.5.3.7.14/14-67
0x2_46B8	RXUO—Receive unknown OP code counter register	R/W	0x0000_0000	14.5.3.7.15/14-68
0x2_46BC	RALN—Receive alignment error counter register	R/W	0x0000_0000	14.5.3.7.16/14-68
0x2_46C0	RFLR—Receive frame length error counter register	R/W	0x0000_0000	14.5.3.7.17/14-69
0x2_46C4	RCDE—Receive code error counter register	R/W	0x0000_0000	14.5.3.7.18/14-69
0x2_46C8	RCSE—Receive carrier sense error counter register	R/W	0x0000_0000	14.5.3.7.19/14-70
0x2_46CC	RUND—Receive undersize packet counter register	R/W	0x0000_0000	14.5.3.7.20/14-70
0x2_46D0	ROVR—Receive oversize packet counter register	R/W	0x0000_0000	14.5.3.7.21/14-71
0x2_46D4	RFRG—Receive fragments counter register	R/W	0x0000_0000	14.5.3.7.22/14-71

Table 14-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
0x2_46D8	RJBR—Receive jabber counter register	R/W	0x0000_0000	14.5.3.7.23/14-72
0x2_46DC	RDRP—Receive drop register	R/W	0x0000_0000	14.5.3.7.24/14-72
TSEC1 Transmit Counters				
0x2_46E0	TBYT—Transmit byte counter register	R/W	0x0000_0000	14.5.3.7.25/14-73
0x2_46E4	TPKT—Transmit packet counter register	R/W	0x0000_0000	14.5.3.7.26/14-73
0x2_46E8	TMCA—Transmit multicast packet counter register	R/W	0x0000_0000	14.5.3.7.27/14-74
0x2_46EC	TBCA—Transmit broadcast packet counter register	R/W	0x0000_0000	14.5.3.7.28/14-74
0x2_46F0	TXPF—Transmit PAUSE control frame counter register	R/W	0x0000_0000	14.5.3.7.29/14-75
0x2_46F4	TDFR—Transmit deferral packet counter register	R/W	0x0000_0000	14.5.3.7.30/14-75
0x2_46F8	TEDF—Transmit excessive deferral packet counter register	R/W	0x0000_0000	14.5.3.7.31/14-76
0x2_46FC	TSCL—Transmit single collision packet counter register	R/W	0x0000_0000	14.5.3.7.32/14-76
0x2_4700	TMCL—Transmit multiple collision packet counter register	R/W	0x0000_0000	14.5.3.7.33/14-77
0x2_4704	TLCL—Transmit late collision packet counter register	R/W	0x0000_0000	14.5.3.7.34/14-77
0x2_4708	TXCL—Transmit excessive collision packet counter register	R/W	0x0000_0000	14.5.3.7.35/14-78
0x2_470C	TNCL—Transmit total collision counter register	R/W	0x0000_0000	14.5.3.7.36/14-78
0x2_4710	Reserved	R	0x0000_0000	—
0x2_4714	TDRP—Transmit drop frame counter register	R/W	0x0000_0000	14.5.3.7.37/14-79
0x2_4718	TJBR—Transmit jabber frame counter register	R/W	0x0000_0000	14.5.3.7.38/14-79
0x2_471C	TFCS—Transmit FCS error counter register	R/W	0x0000_0000	14.5.3.7.39/14-80
0x2_4720	TXCF—Transmit control frame counter register	R/W	0x0000_0000	14.5.3.7.40/14-80
0x2_4724	TOVR—Transmit oversize frame counter register	R/W	0x0000_0000	14.5.3.7.41/14-81
0x2_4728	TUND—Transmit undersize frame counter register	R/W	0x0000_0000	14.5.3.7.42/14-81
0x2_472C	TFRG—Transmit fragments frame counter register	R/W	0x0000_0000	14.5.3.7.43/14-82
TSEC1 General Registers				
0x2_4730	CAR1—Carry register one	R/W	0x0000_0000	14.5.3.7.44/14-82
0x2_4734	CAR2—Carry register two	R/W	0x0000_0000	14.5.3.7.45/14-84
0x2_4738	CAM1—Carry register one mask register	R/W	0xFE01_FFFF	14.5.3.7.46/14-85
0x2_473C	CAM2—Carry register two mask register	R/W	0x000F_FFFF	14.5.3.7.47/14-86
0x2_4740–0x2_47FC	Reserved	R	0x0000_0000	—

Table 14-3. 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	14.5.3.8.1/14-88
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	14.5.3.8.2/14-88
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	14.5.3.9.1/14-89
0x2_4BFC	ATTRELI—Attribute EL & EI register	R	0x0000_0000	14.5.3.9.2/14-90
TSEC1 Future Expansion Space				
0x2_4C00–0x2_4FFC	Reserved	R	0x0000_0000	—

Table 14-3. Module Memory Map (continued)

Offset	Name	Access[1]	Reset	Section/Page
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.

14.5.3 Memory-Mapped Register Descriptions

This section provides a detailed description of all the TSEC registers. Because all of the TSEC registers are 32 bits wide, only 32-bit register accesses are supported.

14.5.3.1 TSEC General Control and Status Registers

This section describes general control and status registers used for both transmitting and receiving Ethernet frames. All of the registers are 32 bits wide.

14.5.3.1.1 Interrupt Event Register (IEVENT)

If an event occurs that sets a bit in the interrupt event (IEVENT) register, shown in Figure 14-6, 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.

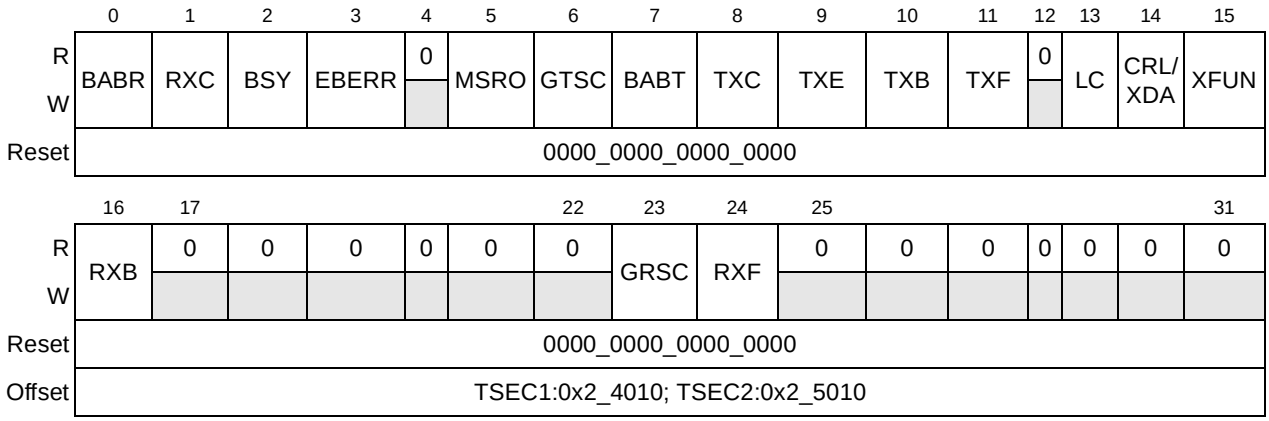


Figure 14-6. IEVENT Register Definition

Table 14-4 describes the fields of the IEVENT register.

Table 14-4. IEVENT Field Descriptions

Bits	Name	Description
0	BABR	Babbling receive error. This bit indicates that a frame was received with length in excess of the MAC's maximum frame length register while MACCFG2[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

Table 14-4. IEVENT Field Descriptions (continued)

Bits	Name	Description
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
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.

Table 14-4. IEVENT Field Descriptions (continued)

Bits	Name	Description
15	XFUN	Transmit FIFO underrun. This bit indicates that the transmit FIFO became empty before the complete frame was transmitted. 0 Transmit FIFO not underrun 1 Transmit FIFO underrun
16	RXB	Receive buffer. 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
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

14.5.3.1.2 Interrupt Mask Register (IMASK)

The interrupt mask register provides control over which possible interrupt events are allowed to generate an actual interrupt. All implemented bits in this CSR are R/W. This register is cleared upon a hardware reset. If the corresponding bits in both the IEVENT and IMASK registers are set, an interrupt is generated. The interrupt signal can be cleared by clearing the corresponding IEVENT bit.

Figure 14-7 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 14-7. IMASK Register Definition

Table 14-5 describes the fields of the IMASK register.

Table 14-5. IMASK Field Descriptions

Bits	Name	Description
0	BREN	Babbling receiver interrupt enable
1	RXCEN	Receive control interrupt enable
2	BSYEN	Busy interrupt enable
3	EBERREN	Ethernet controller bus error enable
4	—	Reserved
5	MSROEN	MSTAT register overflow interrupt enable
6	GTSCEN	Graceful transmit stop complete interrupt enable
7	BTEN	Babbling transmitter interrupt enable
8	TXCEN	Transmit control interrupt enable
9	TXEEN	Transmit error interrupt enable
10	TXBEN	Transmit buffer interrupt enable
11	TXFEN	Transmit frame interrupt enable
12	—	Reserved
13	LCEN	Late collision enable
14	CRL/XDAEN	Collision retry limit/excessive defer enable
15	XFUNEN	Transmit FIFO underrun enable
16	RXBEN	Receive buffer interrupt enable
17–22	—	Reserved
23	GRSCEN	Graceful receive stop complete interrupt enable
24	RXFEN	Receive frame interrupt enable
25–31	—	Reserved

14.5.3.1.3 Error Disabled Register (EDIS)

The error disabled register, shown in Figure 14-8, controls error reporting by the TSEC. The IEVENT bit corresponding to an error will not be set if the error is disabled in EDIS.

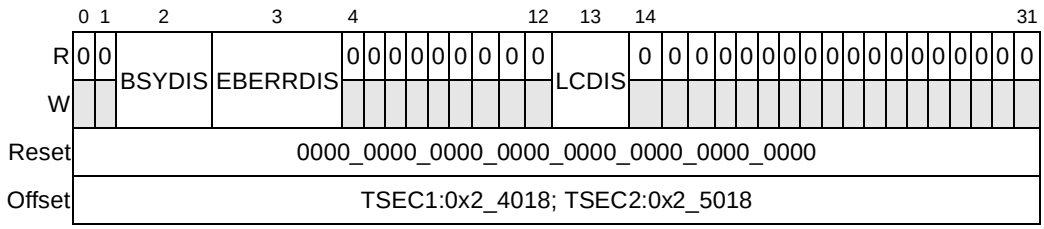


Figure 14-8. Error Disabled Register (EDIS)

Table 14-6 describes the fields of the EDIS register.

Table 14-6. EDIS Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	BSYDIS	Busy disable 0 Allow TSEC to report IEVENT[BSY] status and halt buffer descriptor queue if BSY condition occurs. 1 Do not set IEVENT[BSY] and do not halt buffer descriptor queue if BSY condition occurs.
3	EBERRDIS	Ethernet controller bus error disable 0 Allow TSEC to report IEVENT[EBERR] status and halt buffer descriptor queue if EBERR condition occurs. 1 Do not set IEVENT[EBERR] and do not halt buffer descriptor queue if EBERR condition occurs.
4–12	—	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

14.5.3.1.4 Ethernet Control Register (ECNTRL)

ECNTRL, shown in Figure 14-9, is used to reset, configure, and initialize the TSEC.

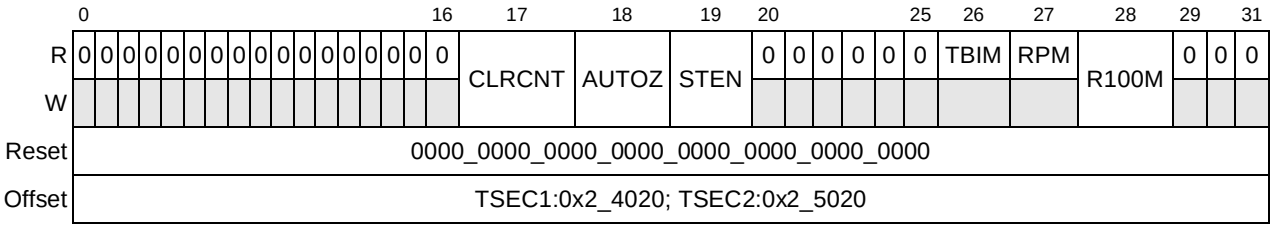


Figure 14-9. ECNTRL Register Definition

Table 14-7 describes the fields of the ECNTRL register.

Table 14-7. ECNTRL Field Descriptions

Bits	Name	Description
0–16	—	Reserved
17	CLRCNT	Clear all statistics counters 0 Allow MSTAT counters to continue to increment. 1 Reset all MSTAT counters. This bit is self-resetting.

Table 14-7. 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

14.5.3.1.5 Minimum Frame Length Register (MINFLR)

MINFLR is written by the user. It tells TSEC the smallest packet to accept and place in a receive buffer pointed to by the RxBD. Figure 14-10 shows the MINFLR register.

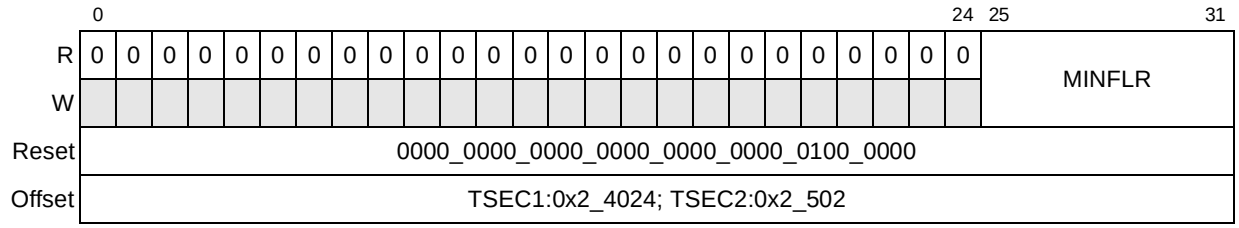


Figure 14-10. MINFLR Register Definition

Table 14-8 describes the fields of the MINFLR register.

Table 14-8. MINFLR Field Descriptions

Bits	Name	Description
0–24	—	Reserved
25–31	MINFLR	Minimum receive frame length (typically 64 decimal). If the Ethernet receives an incoming frame shorter than MINFLR, it discards that frame unless RCTRL[RSF] (receive short frames) is set, in which case RxBD[SH] (frame too short) is set in the last RxBD. The largest allowable value for MINFLR is 64. Unlike the MPC8260, in which PADs are added to make the transmit frame equal to MINFLR bytes, if padding is requested, TSEC always PADS transmit frames to 64 bytes ignoring MINFLR. MINFLR is only used to determine the minimum size of acceptable receive frames.

14.5.3.1.6 Pause Time Value Register (PTV)

PTV, shown in Figure 14-11, 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 14.6.2.7, “Flow Control for additional details.

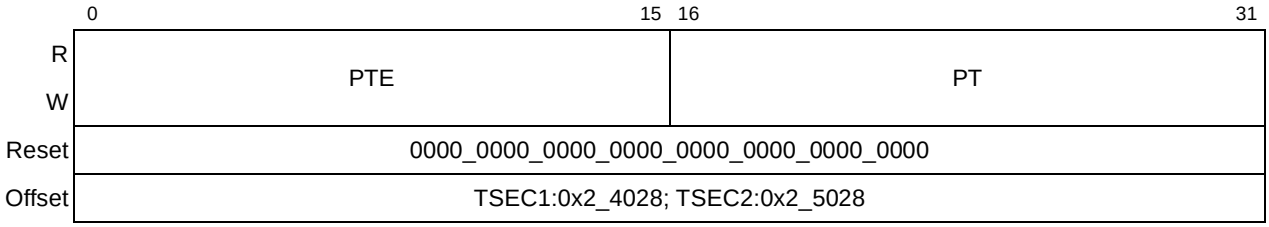


Figure 14-11. PTV Register Definition

Table 14-9 describes the fields of the PTV register.

Table 14-9. PTV Field Descriptions

Bits	Name	Description
0–15	PTE	Extended pause 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 14.6.2.7, “Flow Control,” for more information.

14.5.3.1.7 DMA Control Register (DMACTRL)

DMACTRL, shown in Figure 14-12, is used to configure the DMA block. register.

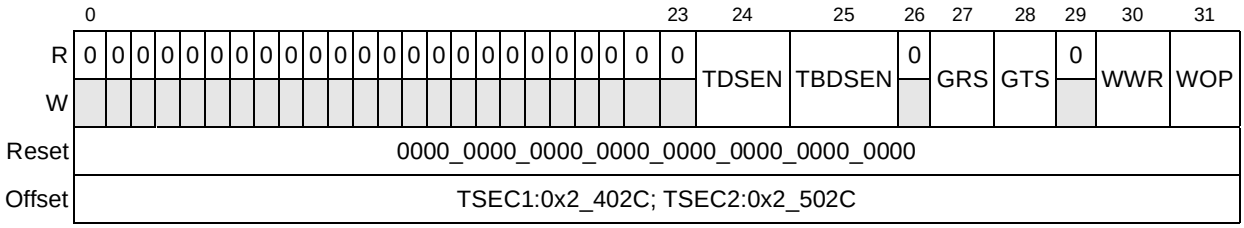


Figure 14-12. DMACTRL Register Definition

Table 14-10 describes the fields of the DMACTRL register.

Table 14-10. DMACTRL Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TDSEN	Tx Data snoop enable 0 Disables snooping of all transmit frames from memory. 1 Enables snooping of all transmit frames from memory.
25	TBDSEN	TxBD snoop enable 0 Disables snooping of all transmit BD memory accesses. 1 Enables snooping of all transmit BD memory accesses.
26	—	Reserved
27	GRS	Graceful receive stop. If this bit is set, the Ethernet controller stops receiving frames following completion of the frame currently being received, that is, after a valid end of frame was received. The buffer of the receive frame associated with the EOF is closed and the IEVENT[GRSC] is set. Because the 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 14-10. 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].

14.5.3.1.8 TBI Physical Address Register (TBIPA)

This TBIPA, shown in Figure 14-13, 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 14.5.4, “Ten-Bit Interface (TBI).”

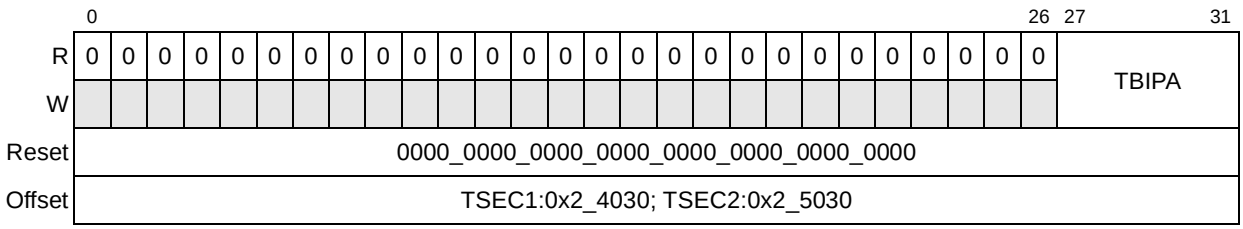


Figure 14-13. TBIPA Register Definition

Table 14-11 describes the fields of the TBIPA register.

Table 14-11. 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 14.5.3.6.8, "MII Management Address Register (MIIMADD)."

14.5.3.2 TSEC FIFO Control and Status Registers

The following registers allow the user to change some of the default settings in the FIFO that can be used to optimize operation for performance or for safety. They must be set carefully in order to avoid an underrun condition. Underrun is an error condition in which data is not retrieved from external memory quickly enough, leaving the TX FIFO empty before the complete frame is transmitted. Because different combinations of events, several of which are determined by the user, can lead to underrun, the TSEC provides several FIFO registers that allow the user to select the proper setting to be able to tune the system and obtain the maximum performance with minimal chance of underrun. The principal causes for underrun in the TSEC are:

- Misaligned data buffer addresses
- Small data buffer sizes
- Combinations of the above

It is recommended that the minimum size data buffers be 64 bytes and that data buffers be 64-byte aligned. The user can deviate from these recommended values to try to increase performance or to use less memory, but unless the default values of some of the FIFO registers are adjusted, the probability of an underrun may also increase. The FIFO_TX_THR (default is 256 entries or 1 Kbytes) indicates the amount of data required to be in the FIFO before starting the transmission of a frame. The FIFO_TX_STARVE (default is 128 entries or 512 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.

14.5.3.2.1 FIFO Pause Control Register (FIFO_PAUSE_CTRL)

FIFO_PAUSE_CTRL, shown in Figure 14-14, is writable by the user to configure the properties of the TSEC FIFO.

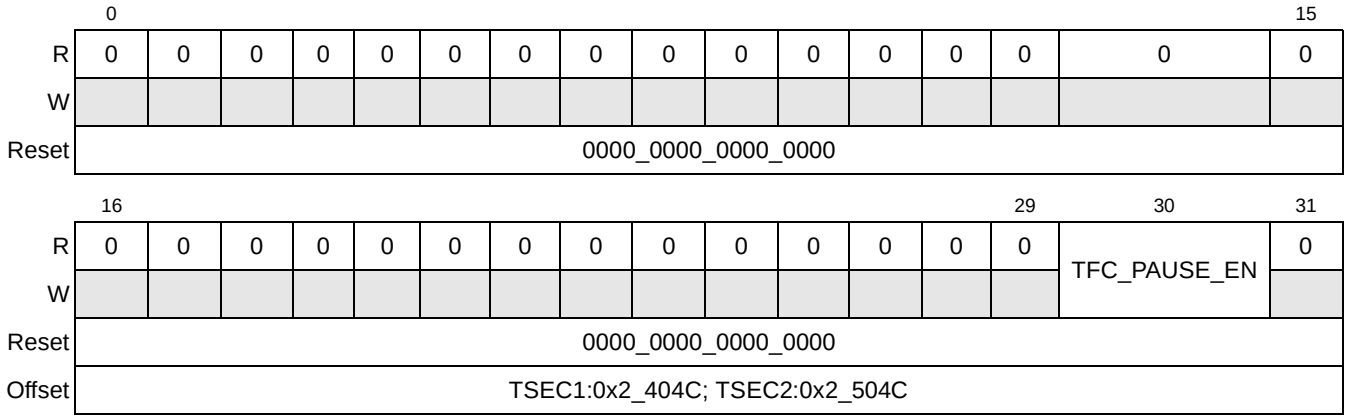


Figure 14-14. FIFO_PAUSE_CTRL Register Definition

Table 14-12 describes the fields of the FIFO_PAUSE_CTRL register.

Table 14-12. 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

14.5.3.2.2 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 14-15 shows the FIFO_TX_THR register.

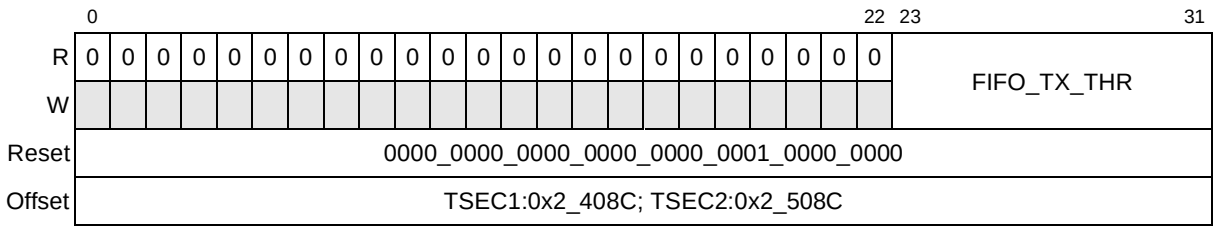


Figure 14-15. FIFO_TX_THR Register Definition

Table 14-13 describes the fields of the FIFO_TX_THR register.

Table 14-13. 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.

14.5.3.2.3 FIFO Transmit Starve Register (FIFO_TX_STARVE)

The purpose of the starve register, shown in Figure 14-16, 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.

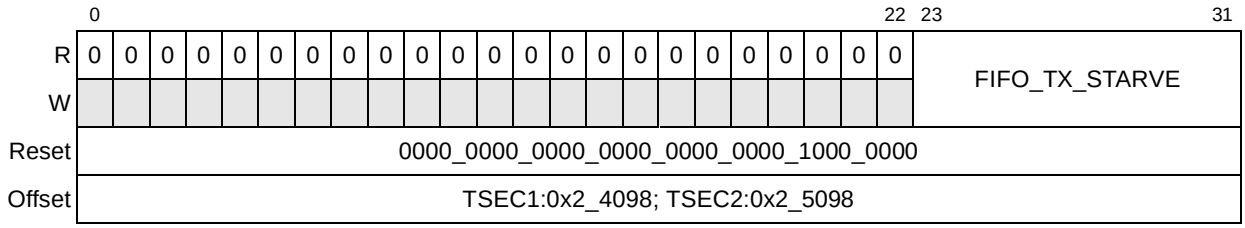


Figure 14-16. FIFO_TX_STARVE Register Definition

Table 14-14 describes the fields of the FIFO_TX_STARVE register.

Table 14-14. 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.

14.5.3.2.4 FIFO Transmit Starve Shutoff Register (FIFO_TX_STARVE_SHUTOFF)

The starve shutoff register, shown in Figure 14-17, 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.

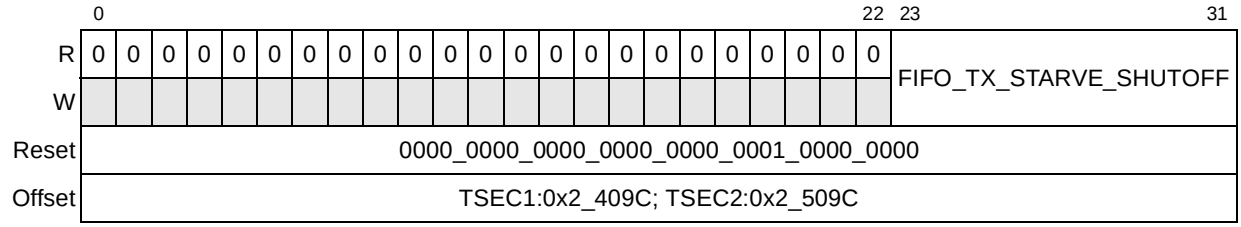


Figure 14-17. FIFO_TX_STARVE_SHUTOFF Register Definition

Table 14-15 describes the fields of the FIFO transmit starve shutoff register.

Table 14-15. 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.

14.5.3.3 TSEC Transmit Control and Status Registers

This section describes the control and status registers that are used specifically for transmitting Ethernet frames. All of the registers are 32 bits wide.

14.5.3.3.1 Transmit Control Register (TCTRL)

TCTRL, shown in Figure 14-18, is writable by the user to configure the transmit block.

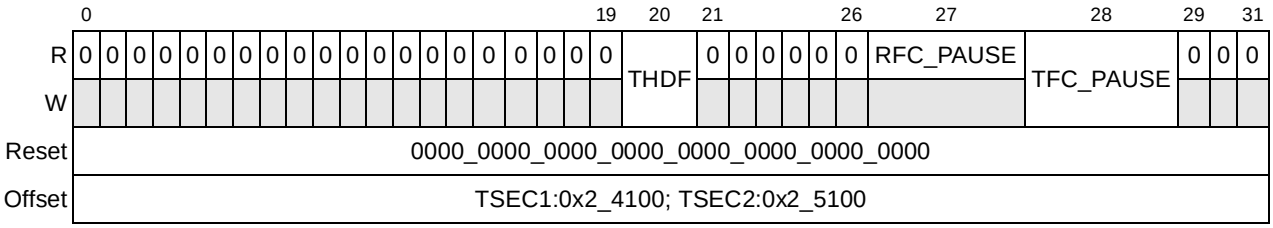


Figure 14-18. TCTRL Register Definition

Table 14-16 describes the fields of the TCTRL register.

Table 14-16. 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 14-16. 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

14.5.3.3.2 Transmit Status Register (TSTAT)

TSTAT, shown in Figure 14-19, is written by TSEC to convey DMA status information.

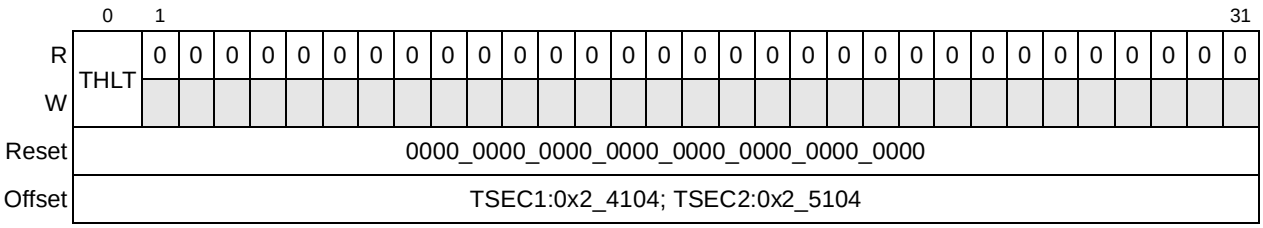


Figure 14-19. TSTAT Register Definition

Table 14-17 describes the fields of the TSTAT register.

Table 14-17. 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

14.5.3.3.3 TxBD Data Length Register (TBDLEN)

TBDLEN is a DMA register that contains the number of bytes remaining in the current transmit buffer. Figure 14-20 shows the TBDLEN register.

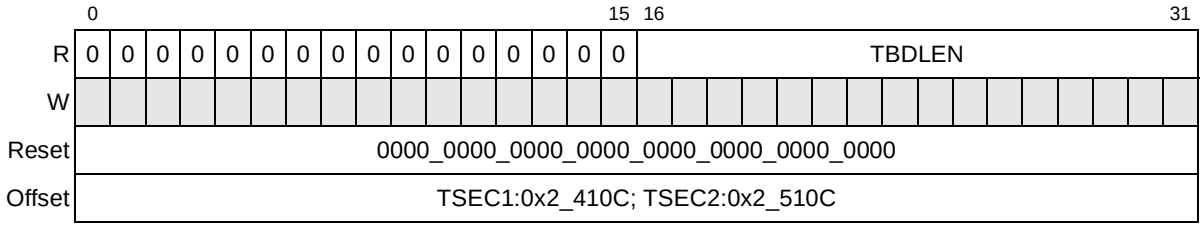


Figure 14-20. TBDLEN Register Definition

Table 14-18 describes the fields of the TBDLEN register.

Table 14-18. 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.

14.5.3.3.4 Transmit Interrupt Coalescing Configuration Register (TXIC)

TXIC, shown in Figure 14-21, enables and configures the operational parameters for interrupt coalescing associated with transmitted frames. Refer to Section 14.6.2.8.1, “Interrupt Coalescing,” for a functional description of interrupt coalescing.

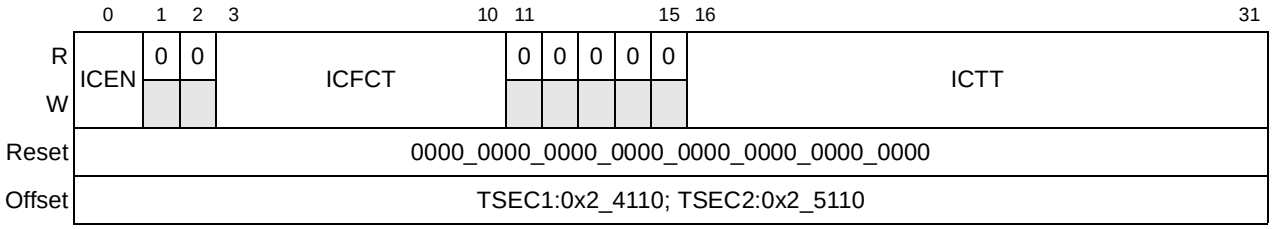


Figure 14-21. TXIC Register Definition

Table 14-18 describes the fields of the TXIC register.

Table 14-19. 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).

14.5.3.3.5 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 14-22 shows the CTBPTR register.

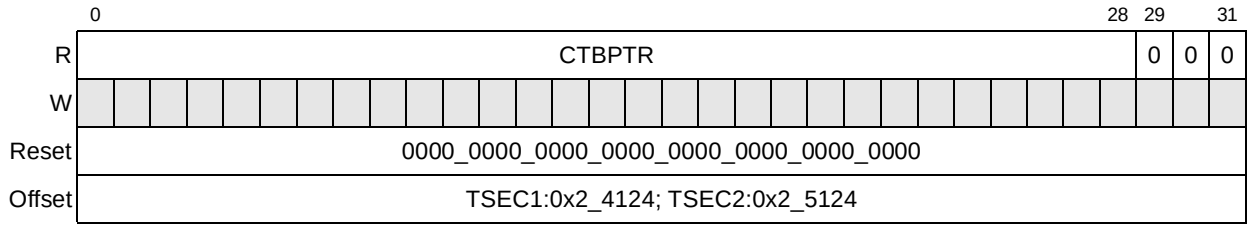


Figure 14-22. CTBPTR Register Definition

Table 14-20 describes the fields of the CTBPTR register.

Table 14-20. 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

14.5.3.3.6 Transmit Buffer Descriptor Pointer Register (TBPTR)

TBPTR, shown in Figure 14-23, 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].

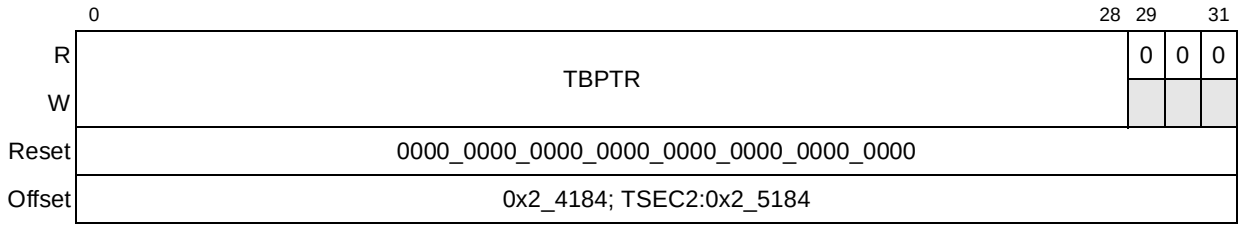


Figure 14-23. TBPTR Register Definition

Table 14-21 describes the fields of the TBPTR register.

Table 14-21. 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

14.5.3.3.7 Transmit Descriptor Base Address Register (TBASE)

The TBASE register, shown in Figure 14-24, is written by the user with the TxBD base address. The value must be divisible by eight for the 8-byte data buffer descriptors.

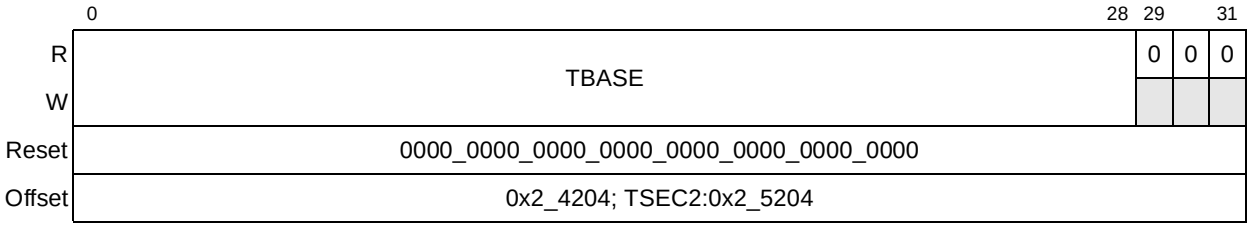


Figure 14-24. TBASE Register Definition

Table 14-22 describes the fields of the TBASE register.

Table 14-22. 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

14.5.3.3.8 Out-of-Sequence TxBD Register (OSTBD)

The out-of-sequence TxBD register, OSTBD, shown in Figure 14-25, 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.

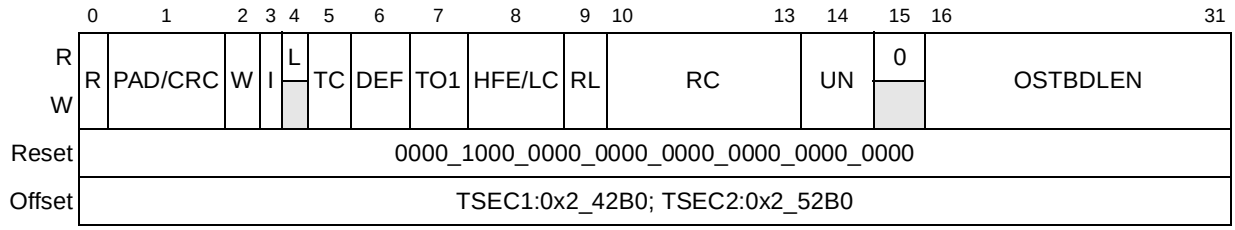


Figure 14-25. OSTBD Register Definition

Table 14-23 describes the fields of the OSTBD register.

Table 14-23. 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 14-23. 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.

14.5.3.3.9 Out-of-Sequence Tx Data Buffer Pointer Register (OSTBDP)

The out-of-sequence Tx data buffer pointer register (OSTBDP), shown in Figure 14-26, 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.

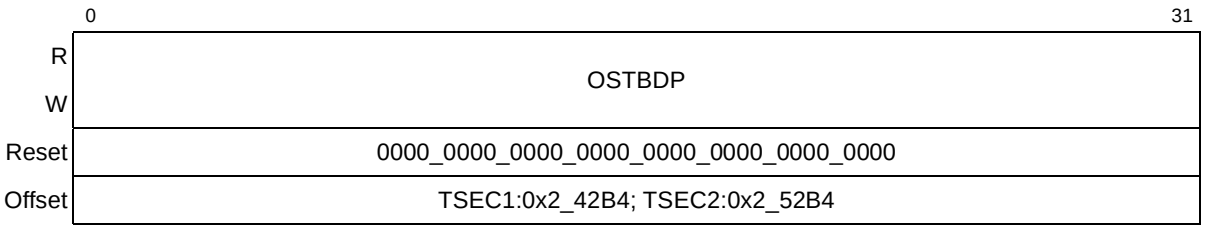


Figure 14-26. OSTBDP Register Definition

Table 14-24 describes OSTBDP.

Table 14-24. 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.

14.5.3.4 TSEC Receive Control and Status Registers

This section describes the control and status registers that are used specifically for receiving Ethernet frames. All of the registers are 32 bits wide.

14.5.3.4.1 Receive Control Register (RCTRL)

RCTRL, shown in Figure 14-27, 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.

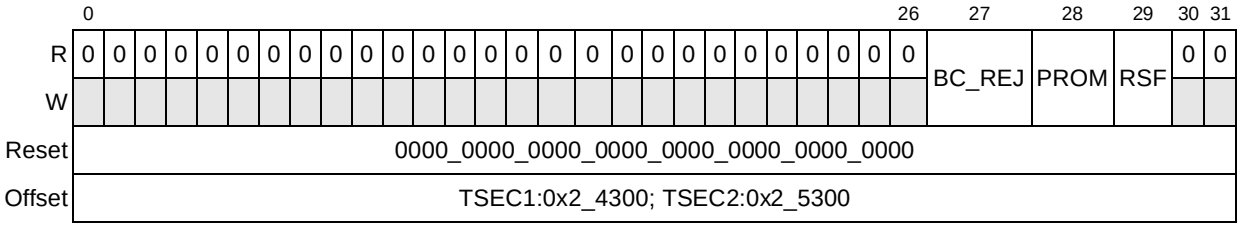


Figure 14-27. RCTRL Register Definition

Table 14-25 describes the fields of the RCTRL register.

Table 14-25. 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

14.5.3.4.2 Receive Status Register (RSTAT)

TSEC writes to RSTAT, shown in Figure 14-28, 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.

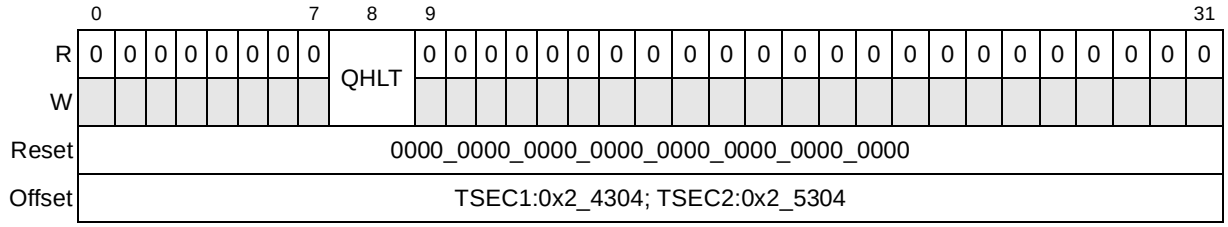


Figure 14-28. RSTAT Register Definition

Table 14-26 describes the fields of the RSTAT register.

Table 14-26. 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

14.5.3.4.3 RxBD Data Length Register (RBDLEN)

RBDLEN is a DMA register that contains the number of bytes remaining in the current receive buffer. Figure 14-29 shows the RBDLEN register.

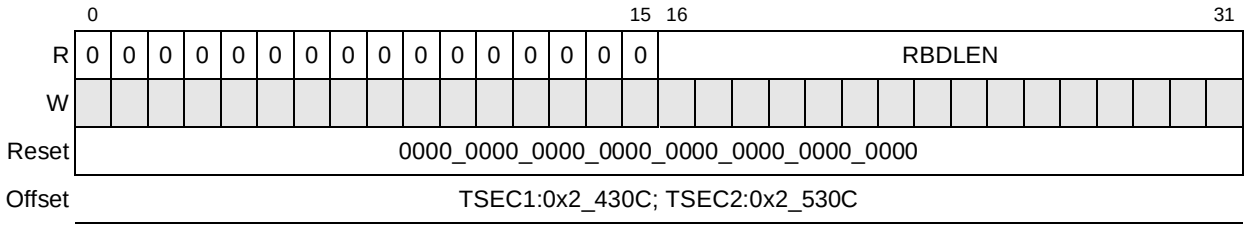


Figure 14-29. RBDLEN Register Definition

Table 14-27 describes the fields of the RBDLEN register.

Table 14-27. 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.

14.5.3.4.4 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 14.6.2.8.1, "Interrupt Coalescing," for a functional description of interrupt coalescing as additional details regarding the use of this register. Figure 14-30 shows the RXIC register.

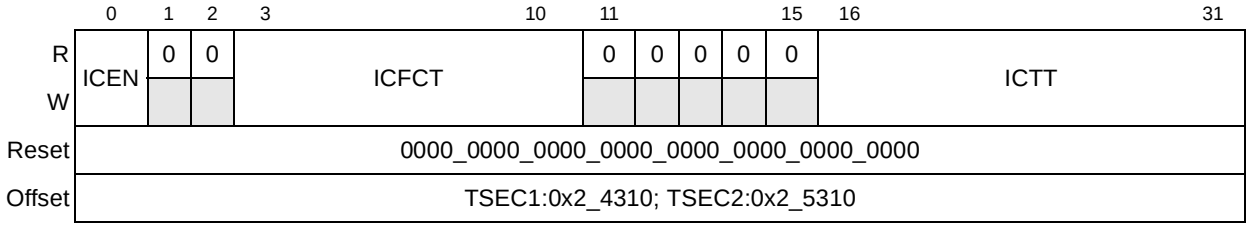


Figure 14-30. RXIC Register Definition

Table 14-18 describes the fields of the RXIC register.

Table 14-28. 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).

14.5.3.4.5 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 14-31 shows the CRBPTR register.

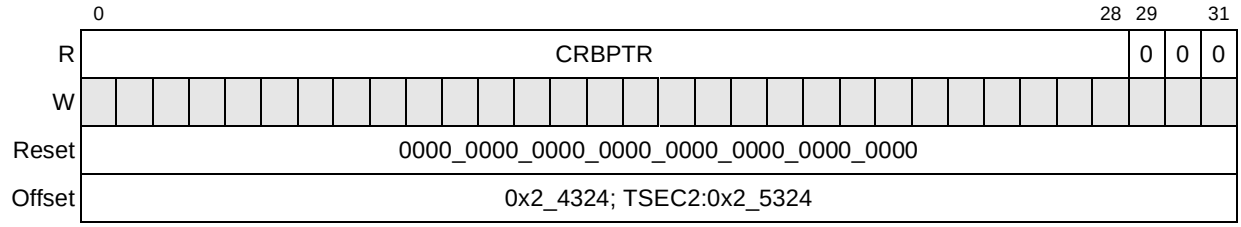


Figure 14-31. CRBPTR Register Definition

Table 14-29 describes the fields of the CRBPTR register.

Table 14-29. 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

14.5.3.4.6 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 14-32 shows the MRBLR.

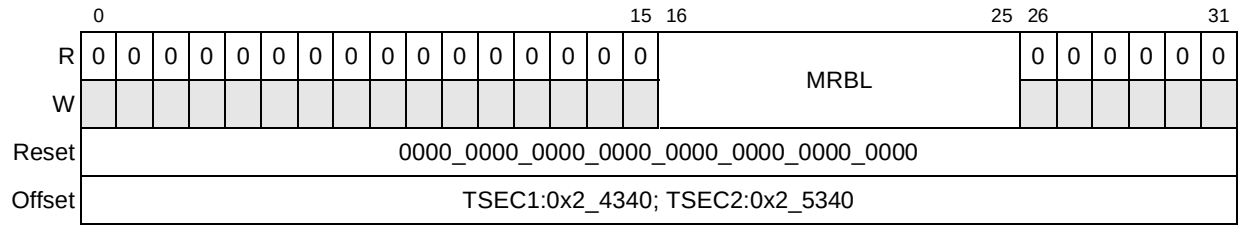


Figure 14-32. MRBL Register Definition

Table 14-30 describes the fields of the MRBL register.

Table 14-30. 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.

14.5.3.4.7 Receive Buffer Descriptor Pointer Register (RBPTR)

RBPTR contains the receive buffer descriptor address. Figure 14-33 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].

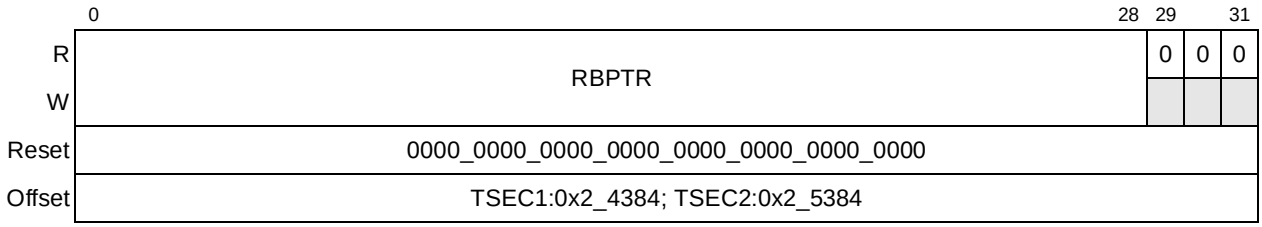


Figure 14-33. RBPTR Register Definition

Table 14-31 describes the fields of the RBPTR register.

Table 14-31. 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

14.5.3.4.8 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 14-34 shows the RBASE register.

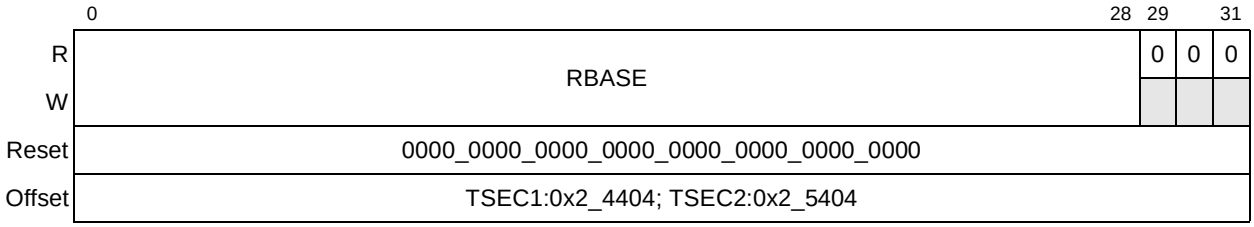


Figure 14-34. RBASE Register Definition

Table 14-32 describes the fields of the RBASE register.

Table 14-32. 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.

14.5.3.5 MAC Functionality

This section describes the MAC registers and provides a brief overview of the functionality that can be exercised through the use of these registers, particularly those that provide functionality not explicitly required by the IEEE 802.3 standard. All of the MAC registers are 32 bits wide.

14.5.3.5.1 Configuring the MAC

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.

14.5.3.5.2 Controlling CSMA/CD

The half-duplex register (HAFDUP) allows control over the carrier-sense multiple access/collision detection (CSMA/CD) logic of the TSEC. Half-duplex is only supported for 10 Mbps and 100 Mbps operation. Following the completion of the packet transmission the part begins timing the inter packet gap (IPG) as programmed in the back-to-back IPG configuration register. The system is now free to begin another frame transfer.

In full-duplex mode both the carrier sense (CRS) and collision (COL) indications from the PHY are ignored, but in half-duplex mode the TSEC defers to CRS and, following a carrier event, times the IPG using the non-back-to-back IPG configuration values that include support for the optional two-thirds/one-third CRS deferral process. This optional IPG mechanism enhances system robustness and ensures fair access to the medium. During the first two-thirds of the IPG, the IPG timer is cleared if CRS is sensed. During the final one-third of the IPG, CRS is ignored and the transmission begins once IPG is timed. The two-thirds/one-third ratio is the recommended value.

14.5.3.5.3 Handling Packet Collisions

While transmitting a packet in half-duplex mode, the TSEC is sensitive to COL. If a collision occurs, it aborts the packet and outputs the 32-bit jam sequence. The jam sequence is comprised of several bits of the CRC, inverted to guarantee an invalid CRC upon reception. A signal is sent to the system indicating that a collision occurred and that the start of the frame is needed for retransmission. The TSEC then backs off of the medium for a time determined by the truncated binary exponential back-off (BEB) algorithm. Following this back-off time, the packet is retried. The back-off time can be skipped if configured via the half-duplex register. However, this is non-standard behavior and its use must be carefully applied. Should any one packet experience excessive collisions, the packet is aborted. The system 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.

14.5.3.5.4 Controlling Packet Flow

Packet flow can be dealt with in a number of ways within TSEC. A default retransmit attempt limit of 15 can be reduced using the half-duplex register. The slot time or collision window can be used to gate the retry window and possibly reduce the amount of transmit buffering within the system. The slot time for 10/100 Mbps is 512 bit times. Because the slot time begins at the beginning of

the packet, the end occurs around the 56th byte of the frame data. Slot time in 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.

14.5.3.5.5 Controlling PHY Links

Control and status to and from the PHY is provided via the two-wire MII management interface described in IEEE 802.3u. The MII management registers (MII management configuration, command, address, control, status, and indicator registers) are used to exercise this interface between a host processor and one or more PHY devices (including the TBI). 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.

14.5.3.6 MAC Registers

The following are the MAC registers.

14.5.3.6.1 MAC Configuration Register 1 (MACCFG1)

The MACCFG1 register is written by the user. Figure 14-35 shows the MACCFG1 register.

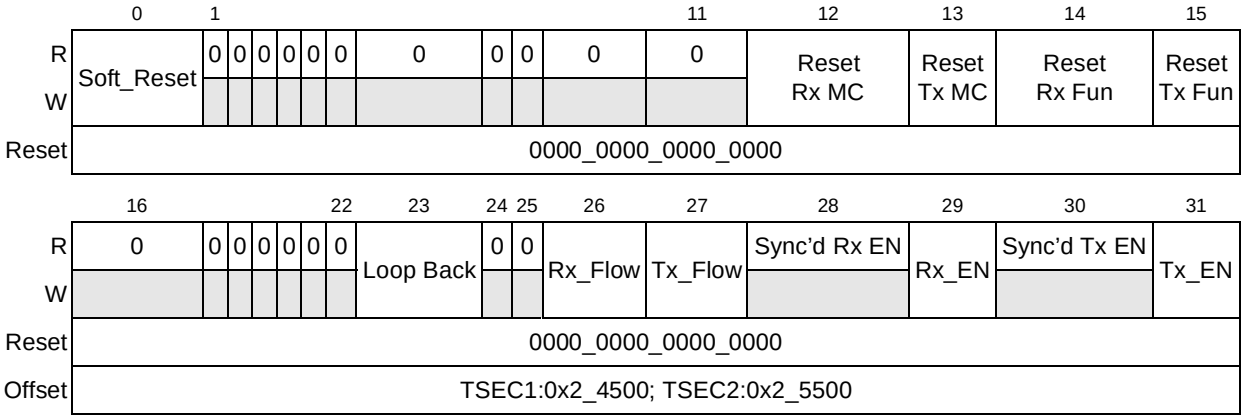


Figure 14-35. MACCFG1 Register Definition

Table 14-33 describes the fields of the MACCFG1 register.

Table 14-33. MACCFG1 Field Descriptions

Bits	Name	Description
0	Soft_Reset	Soft reset. This bit is cleared by default. See Section 14.6.2.2, "Soft Reset and Reconfiguring Procedure," 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 14-33. 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.

14.5.3.6.2 MAC Configuration Register 2 (MACCFG2)

The MACCFG2 register is written by the user. Figure 14-36 shows the MACCFG2 register.

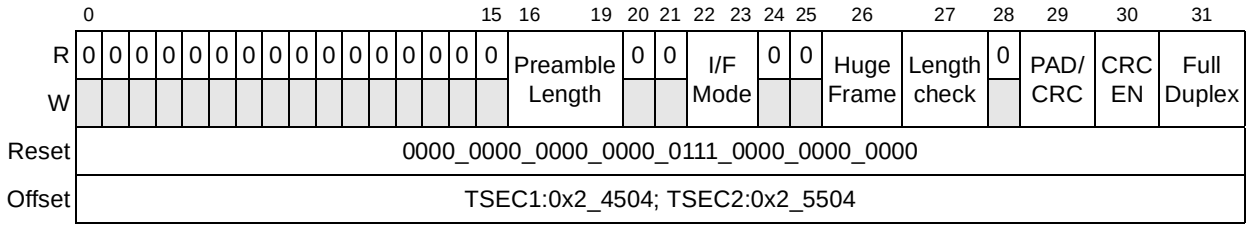


Figure 14-36. MACCFG2 Register Definition

Table 14-34 describes the fields of the MACCFG2 register.

Table 14-34. 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.

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

The IPGIFG register is written by the user. Figure 14-37 shows the IPGIFG register.

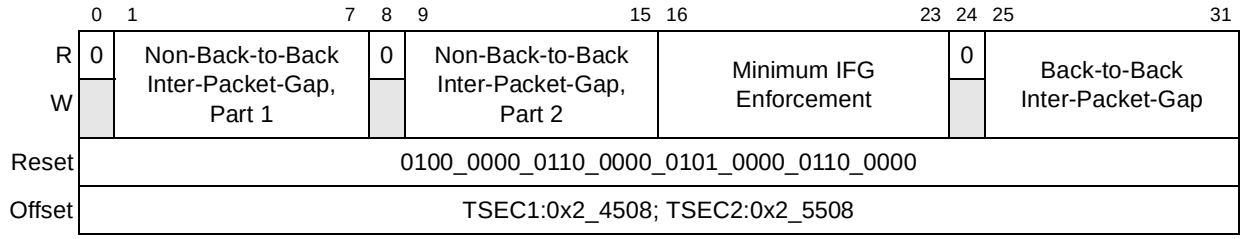


Figure 14-37. IPGIFG Register Definition

Table 14-35 describes the fields of the IPGIFG register.

Table 14-35. 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.

14.5.3.6.4 Half-Duplex Register (HAFDUP)

The HAFDUP register is written by the user. Figure 14-38 shows the HAFDUP register.

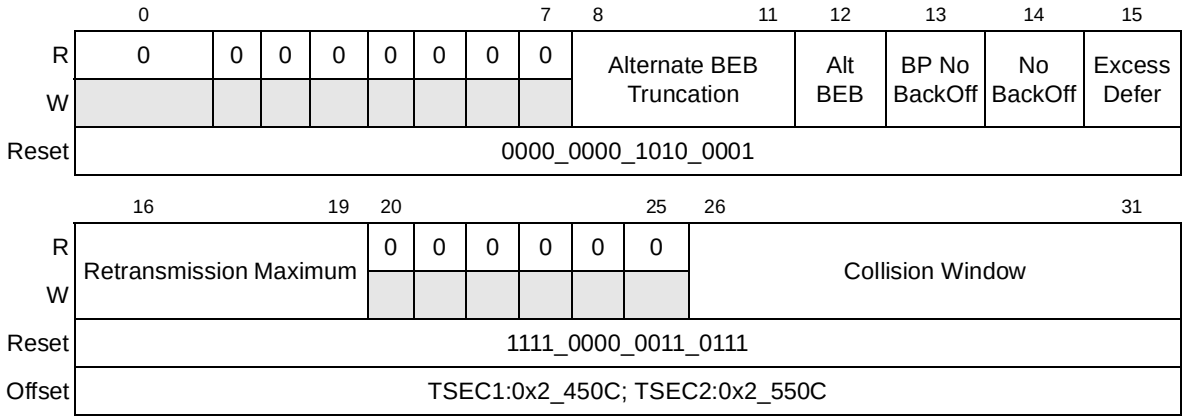


Figure 14-38. Half-Duplex Register Definition

Table 14-36 describes the fields of the HAFDUP register.

Table 14-36. 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 14-36. 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.

14.5.3.6.5 Maximum Frame Length Register (MAXFRM)

The MAXFRM register is written by the user. Figure 14-39 shows the MAXFRM register.

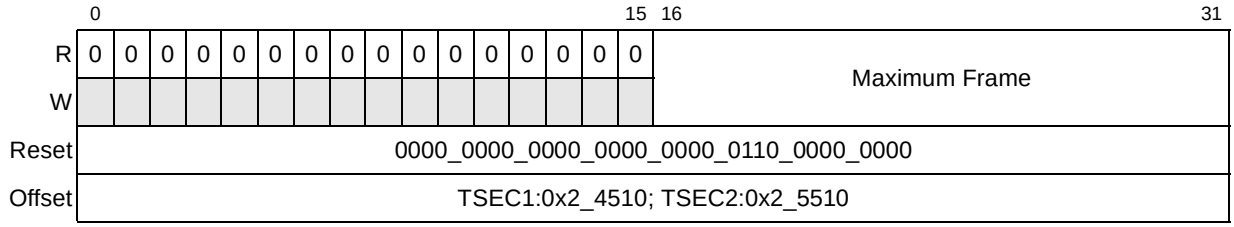


Figure 14-39. Maximum Frame Length Register Definition

Table 14-37 describes the fields of the MAXFRM register.

Table 14-37. 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].)

14.5.3.6.6 MII Management Configuration Register (MIIMCFG)

The MIIMCFG register is written by the user. Figure 14-40 shows the MIIMCFG register.

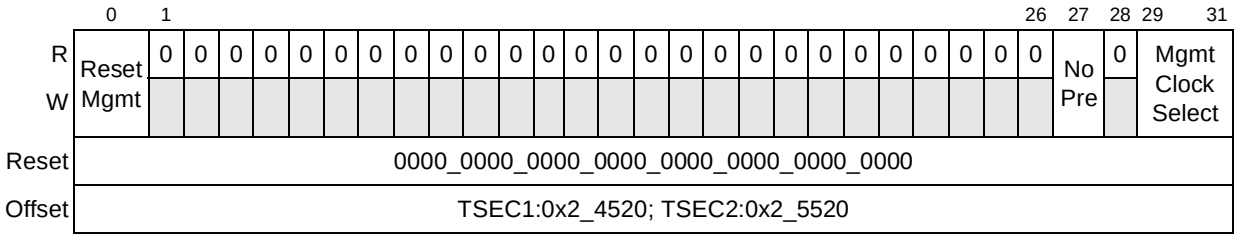


Figure 14-40. MII Management Configuration Register Definition

Table 14-38 describes the fields of the MIIMCFG register.

Table 14-38. 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

14.5.3.6.7 MII Management Command Register (MIIMCOM)

The MIIMCOM register is written by the user. Figure 14-41 shows the MIIMCOM register.

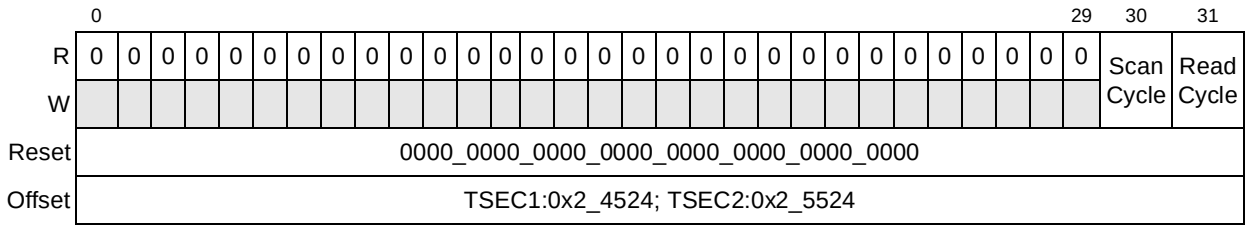


Figure 14-41. MIIMCOM Register Definition

Table 14-39 describes the fields of the MIIMCOM register.

Table 14-39. 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].

14.5.3.6.8 MII Management Address Register (MIIMADD)

The MIIMADD register is written by the user. Figure 14-42 shows the MIIMADD register.

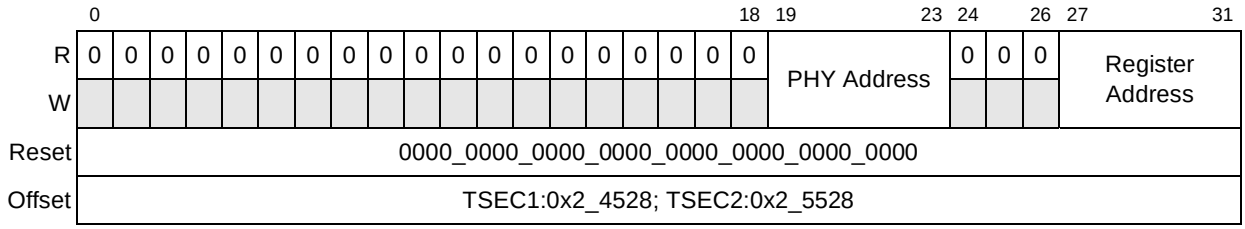


Figure 14-42. MIIMADD Register Definition

Table 14-40 describes the fields of the MIIMADD register.

Table 14-40. 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.

14.5.3.6.9 MII Management Control Register (MIIMCON)

The MIIMCON register is written by the user. Figure 14-43 shows the MIIMCON register.

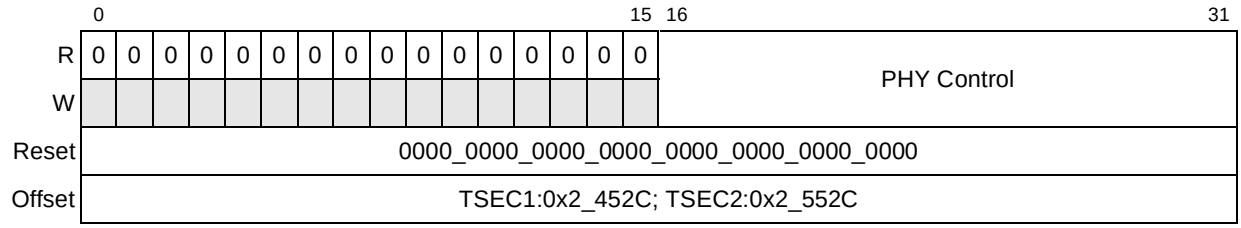


Figure 14-43. MII Management Control Register Definition

Table 14-41 describes the fields of the MIIMCON register.

Table 14-41. 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.

14.5.3.6.10 MII Management Status Register (MIIMSTAT)

The MIIMSTAT, shown in Figure 14-44, is read-only by the user.

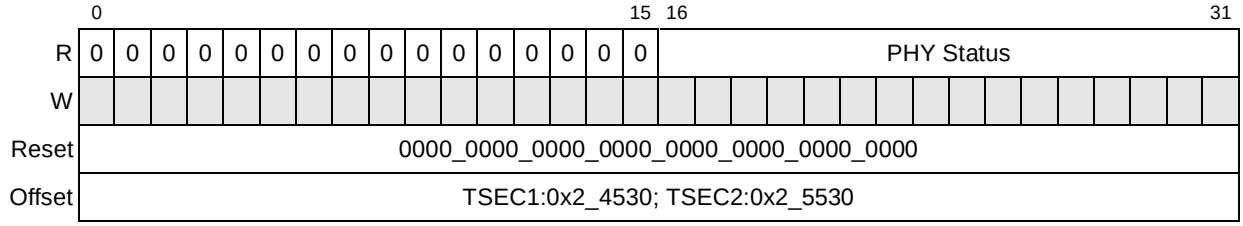


Figure 14-44. MIIMSTAT Register Definition

Table 14-42 describes the fields of the MIIMSTAT register.

Table 14-42. 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.

14.5.3.6.11 MII Management Indicator Register (MIIMIND)

The MIIMIND register is read-only by the user. Figure 14-45 shows the MIIMIND register.

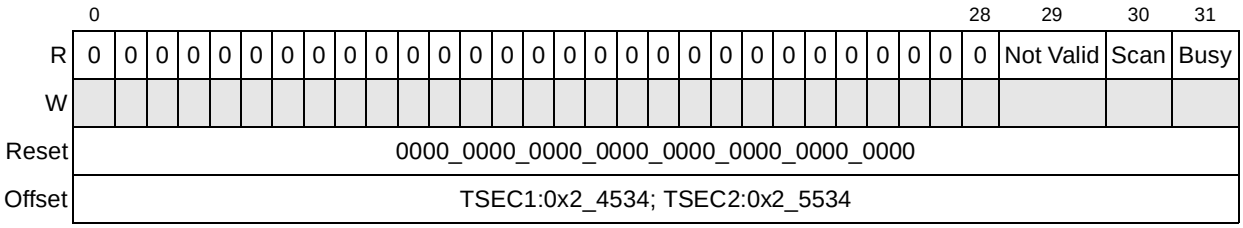


Figure 14-45. MII Management Indicator Register Definition

Table 14-43 describes the fields of the MIIMIND register.

Table 14-43. 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.

14.5.3.6.12 Interface Status Register (IFSTAT)

Figure 14-46 shows the IFSTAT register.

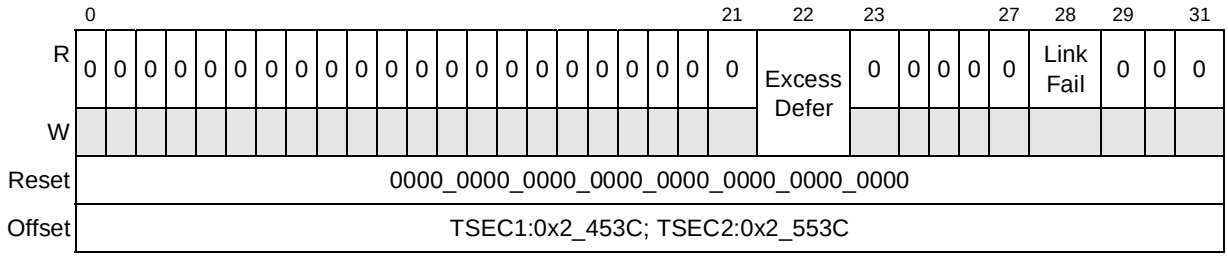


Figure 14-46. Interface Status Register Definition

Table 14-44 describes the fields of the IFSTAT register.

Table 14-44. 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

14.5.3.6.13 Station Address Register Part 1 (MACSTNADDR1)

The MACSTNADDR1 register is written by the user. Figure 14-47 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.

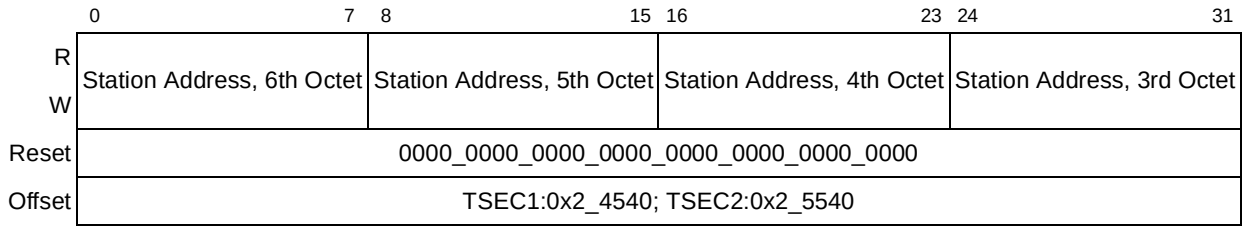


Figure 14-47. Station Address Part 1 Register Definition

Table 14-45 describes the fields of the MACSTNADDR1 register.

Table 14-45. 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.

14.5.3.6.14 Station Address Register Part 2 (MACSTNADDR2)

The MACSTNADDR2 register is written by the user. Figure 14-48 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.

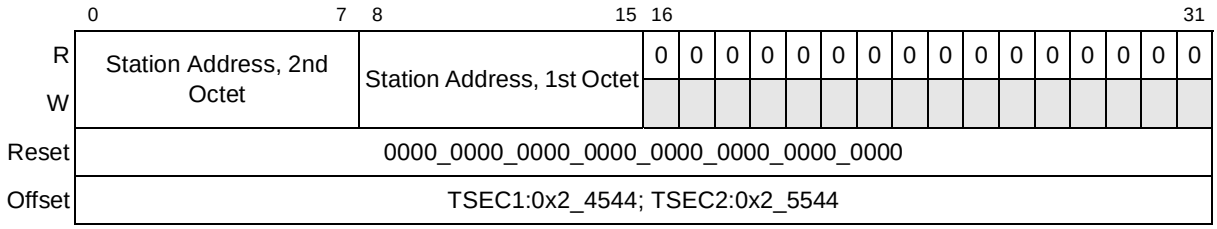


Figure 14-48. Station Address Part 2 Register Definition

Table 14-46 describes the fields of the MACSTNADDR2 register.

Table 14-46. 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

14.5.3.7 MIB Registers

This section describes the MIB registers.

The TSEC MSTAT module has 37 separate statistics counters, which simply count or accumulate statistical events that occur as packets 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 14-49 shows the TR64 register.

14.5.3.7.1 Transmit and Receive 64-Byte Frame Counter Register (TR64)

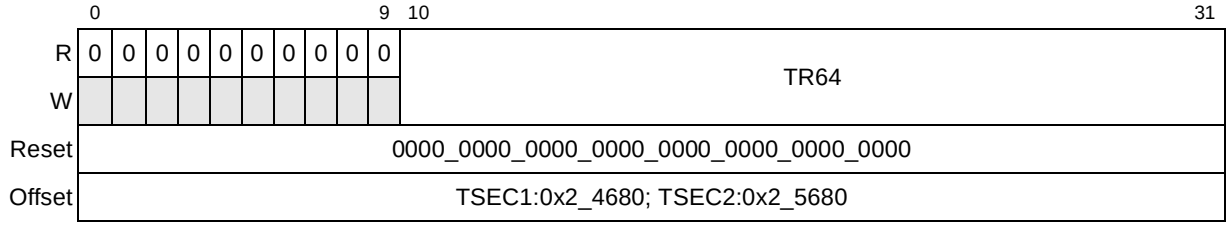


Figure 14-49. Transmit and Receive 64-Byte Frame Register Definition

Table 14-47 describes the fields of the TR64 register.

Table 14-47. 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)

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

Figure 14-50 shows the TR127 register.

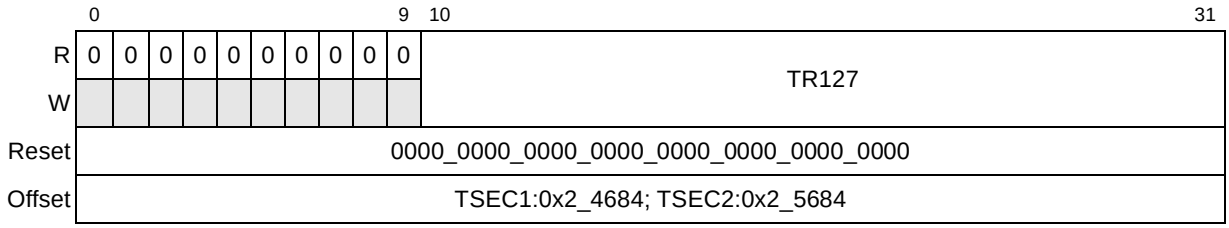


Figure 14-50. Transmit and Receive 65- to 127-Byte Frame Register Definition

Table 14-48 describes the fields of the TR127 register.

Table 14-48. 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)

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

Figure 14-51 shows the TR255 register.

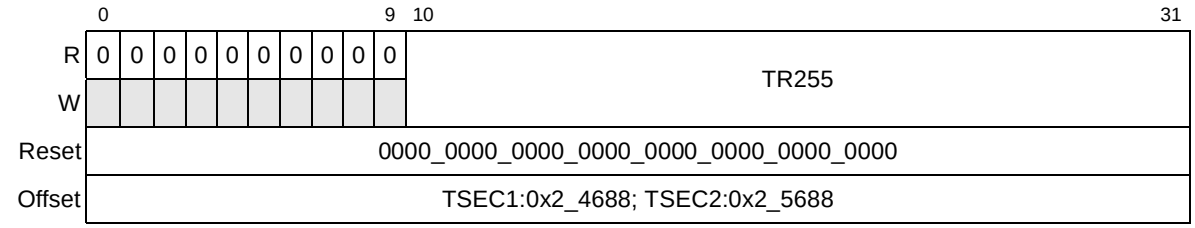


Figure 14-51. Transmit and Receive 128- to 255-Byte Frame Register Definition

Table 14-49 describes the fields of the TR255 register.

Table 14-49. 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)

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

Figure 14-52 shows the TR511 register.

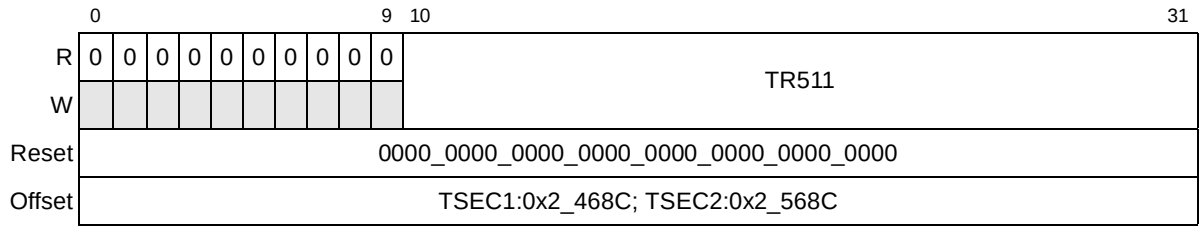


Figure 14-52. Transmit and Receive 256- to 511-Byte Frame Register Definition

Table 14-50 describes the fields of the TR511 register.

Table 14-50. 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)

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

Figure 14-53 shows the TR1K register.

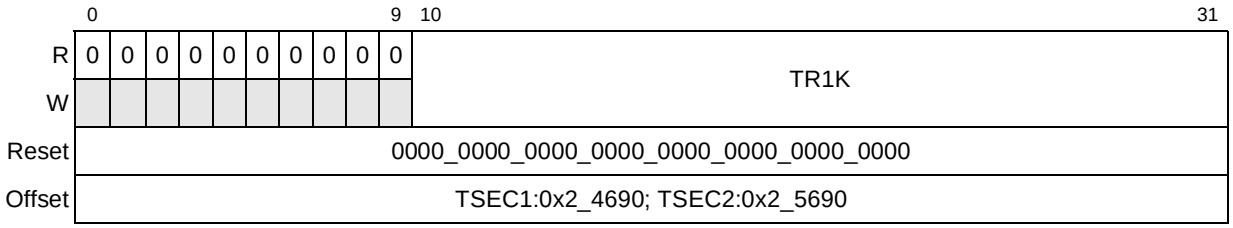


Figure 14-53. Transmit and Receive 512- to 1023-Byte Frame Register Definition

Table 14-51 describes the fields of the TR1K register.

Table 14-51. 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)

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

Figure 14-54 shows the TRMAX register.

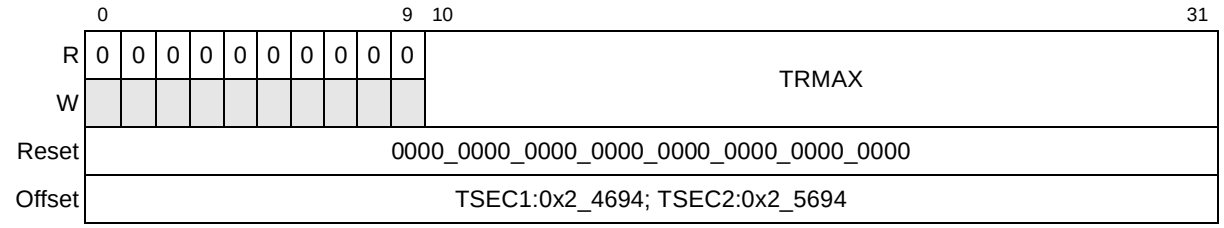


Figure 14-54. Transmit and Receive 1024- to 1518-Byte Frame Register Definition

Table 14-52 describes the fields of the TRMAX register.

Table 14-52. 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)

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

Figure 14-55 shows the TRMGV register.

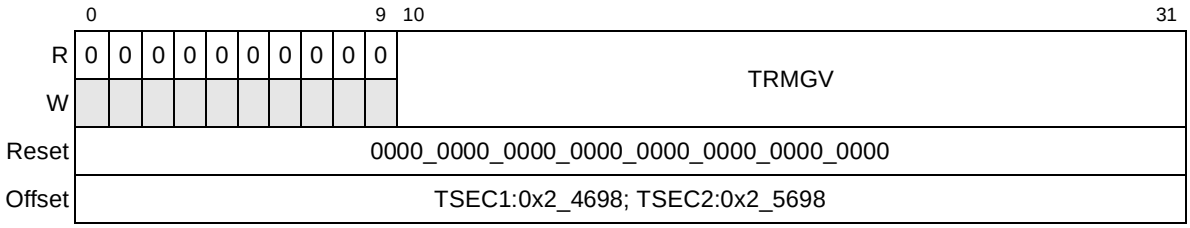


Figure 14-55. Transmit and Receive 1519- to 1522-Byte VLAN Frame Register Definition

Table 14-53 describes the fields of the TRMGV register.

Table 14-53. 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)

14.5.3.7.8 Receive Byte Counter Register (RBYT)

Figure 14-56 shows the RBYT register.

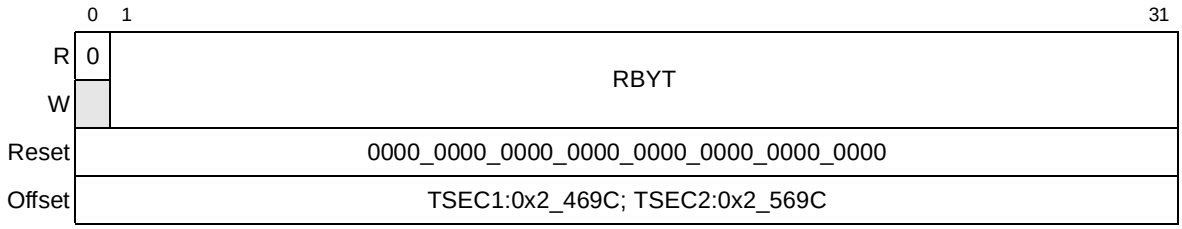


Figure 14-56. Receive Byte Counter Register Definition

Table 14-54 describes the fields of the RBYT register.

Table 14-54. 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.

14.5.3.7.9 Receive Packet Counter Register (RPKT)

Figure 14-57 shows the RPKT register.

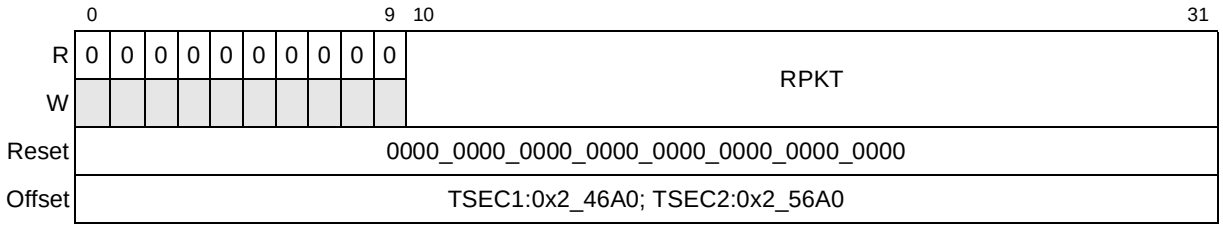


Figure 14-57. Receive Packet Counter Register Definition

Table 14-55 describes the fields of the RPKT register.

Table 14-55. 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).

14.5.3.7.10 Receive FCS Error Counter Register (RFCS)

Figure 14-58 shows the RFCS register.

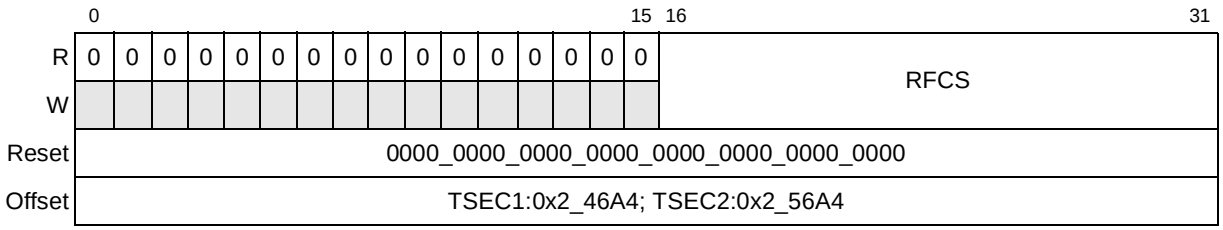


Figure 14-58. Receive FCS Error Counter Register Definition

Table 14-56 describes the fields of the RFCS register.

Table 14-56. 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.

14.5.3.7.11 Receive Multicast Packet Counter Register (RMCA)

Figure 14-59 shows the RMCA register.

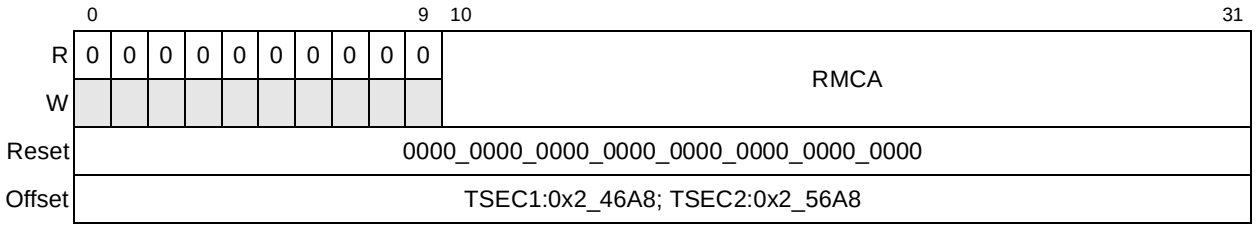


Figure 14-59. Receive Multicast Packet Counter Register Definition

Table 14-57 describes the fields of the RMCA register.

Table 14-57. 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.

14.5.3.7.12 Receive Broadcast Packet Counter Register (RBCA)

Figure 14-59 shows the RBCA register.

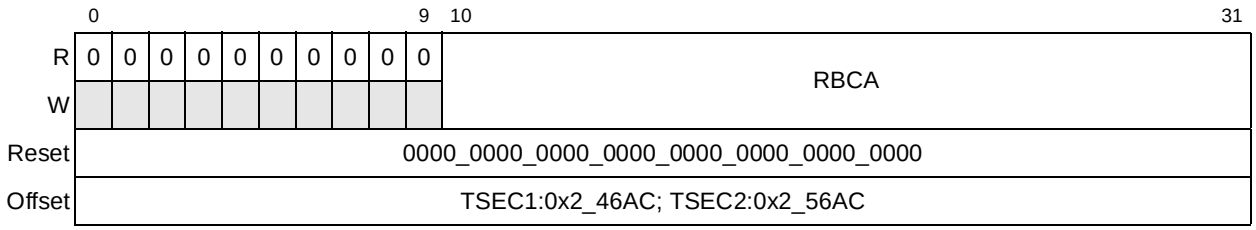


Figure 14-60. Receive Broadcast Packet Counter Register Definition

Table 14-58 describes the fields of the RBCA register.

Table 14-58. 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.

14.5.3.7.13 Receive Control Frame Packet Counter Register (RXCF)

Figure 14-61 shows the RXCF register.

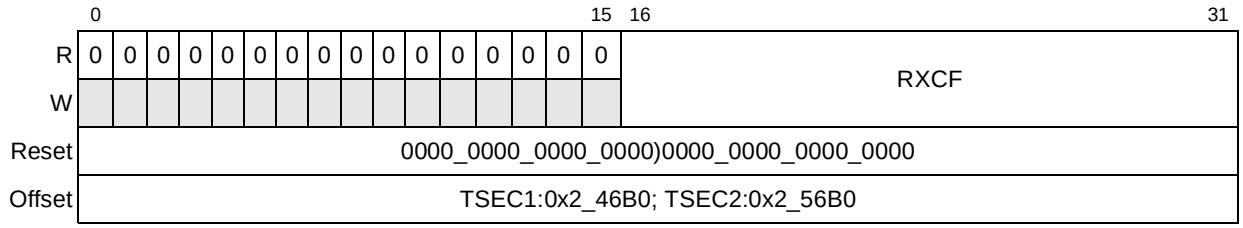


Figure 14-61. Receive Control Frame Packet Counter Register Definition

Table 14-59 describes the fields of the RXCF register.

Table 14-59. 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).

14.5.3.7.14 Receive Pause Frame Packet Counter Register (RXPF)

Figure 14-62 shows the RXPF register.

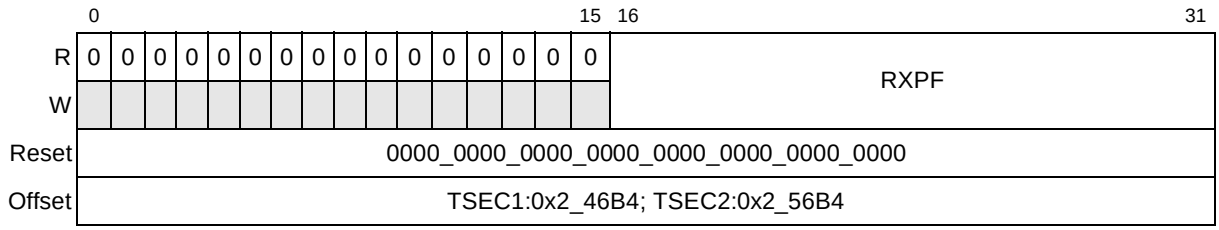


Figure 14-62. Receive Pause Frame Packet Counter Register Definition

Table 14-60 describes the fields of the RXPF register.

Table 14-60. 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.

14.5.3.7.15 Receive Unknown Opcode Packet Counter Register (RXUO)

Figure 14-63 shows the RXUO register.

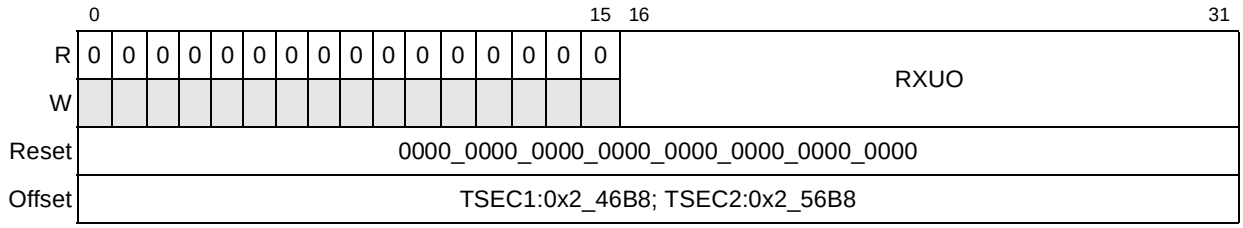


Figure 14-63. Receive Unknown Opcode Packet Counter Register Definition

Table 14-61 describes the fields of the RXUO register.

Table 14-61. 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.

14.5.3.7.16 Receive Alignment Error Counter Register (RALN)

Figure 14-64 shows the RALN register.

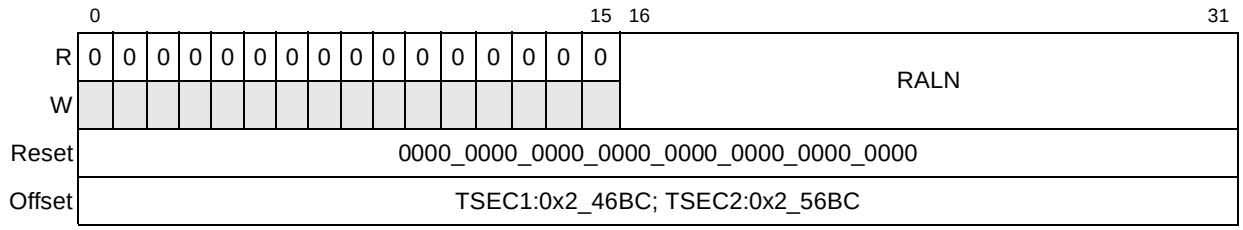


Figure 14-64. Receive Alignment Error Counter Register Definition

Table 14-62 describes the fields of the RALN register.

Table 14-62. 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.

14.5.3.7.17 Receive Frame Length Error Counter Register (RFLR)

Figure 14-65 shows the RFLR register.

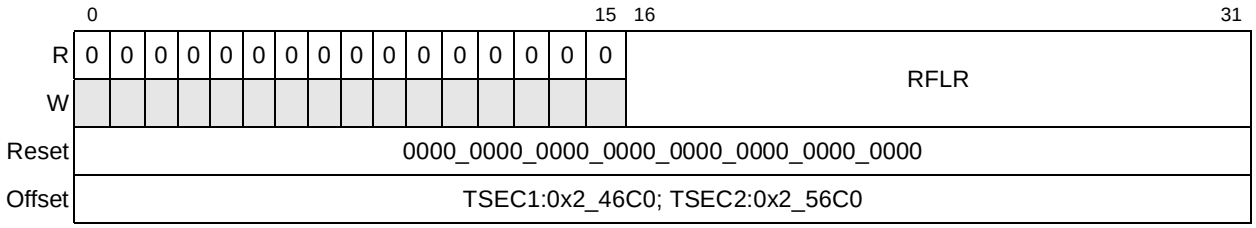


Figure 14-65. Receive Frame Length Error Counter Register Definition

Table 14-63 describes the fields of the RFLR register.

Table 14-63. 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.

14.5.3.7.18 Receive Code Error Counter Register (RCDE)

Figure 14-66 shows the RCDE register.

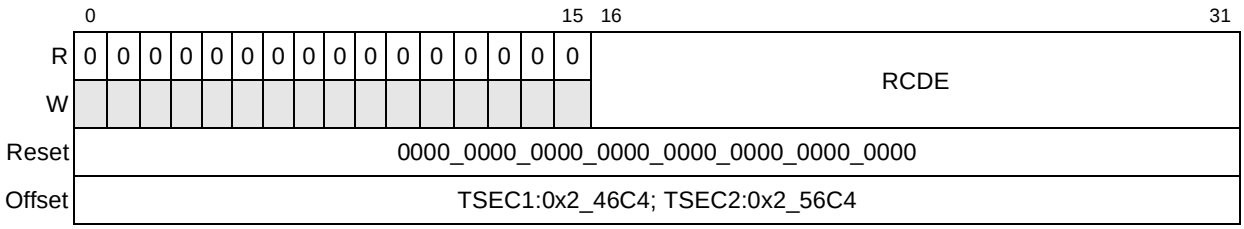


Figure 14-66. Receive Code Error Counter Register Definition

Table 14-64 describes the fields of the RCDE register.

Table 14-64. 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.

14.5.3.7.19 Receive Carrier Sense Error Counter Register (RCSE)

Figure 14-67 shows the RCSE register.

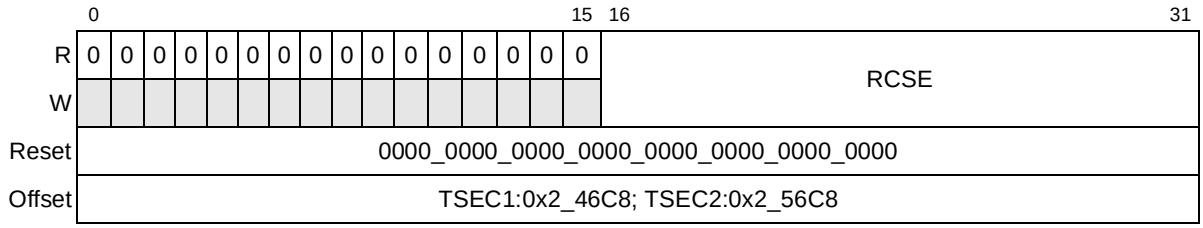


Figure 14-67. Receive Carrier Sense Error Counter Register Definition

Table 14-65 describes the fields of the RCSE register.

Table 14-65. 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.

14.5.3.7.20 Receive Undersize Packet Counter Register (RUND)

Figure 14-68 shows the RUND register.

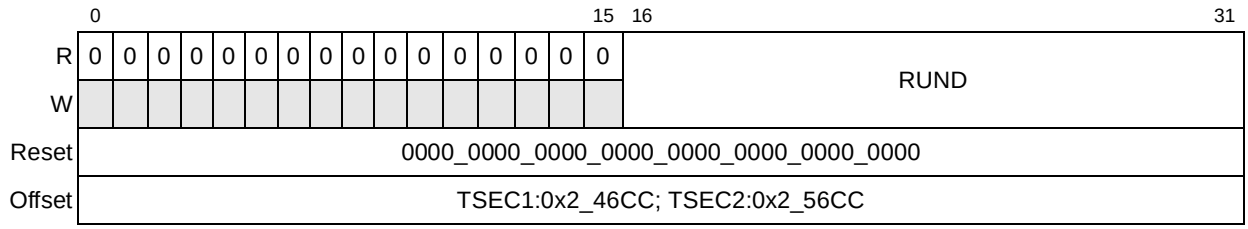


Figure 14-68. Receive Undersize Packet Counter Register Definition

Table 14-66 describes the fields of the RUND register.

Table 14-66. 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.

14.5.3.7.21 Receive Oversize Packet Counter Register (ROVR)

Figure 14-69 shows the ROVR register.

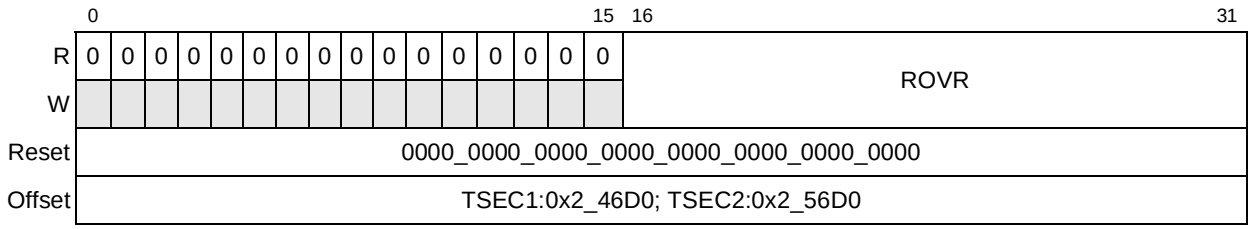


Figure 14-69. Receive Oversize Packet Counter Register Definition

Table 14-67 describes the fields of the ROVR register.

Table 14-67. 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.

14.5.3.7.22 Receive Fragments Counter Register (RFRG)

Figure 14-70 shows the RFRG register.

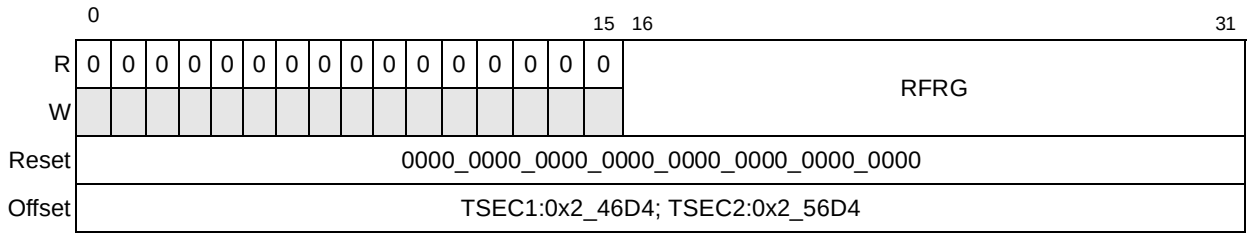


Figure 14-70. Receive Fragments Counter Register Definition

Table 14-68 describes the fields of the RFRG register.

Table 14-68. 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.

14.5.3.7.23 Receive Jabber Counter Register (RJBR)

Figure 14-71 shows the RJBR register.

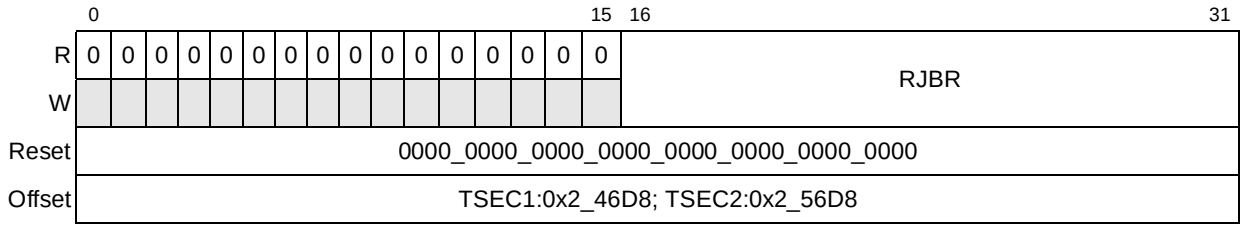


Figure 14-71. Receive Jabber Counter Register Definition

Table 14-69 describes the fields of the RJBR register.

Table 14-69. 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.

14.5.3.7.24 Receive Dropped Packet Counter Register (RDRP)

Figure 14-72 shows the RDRP register.

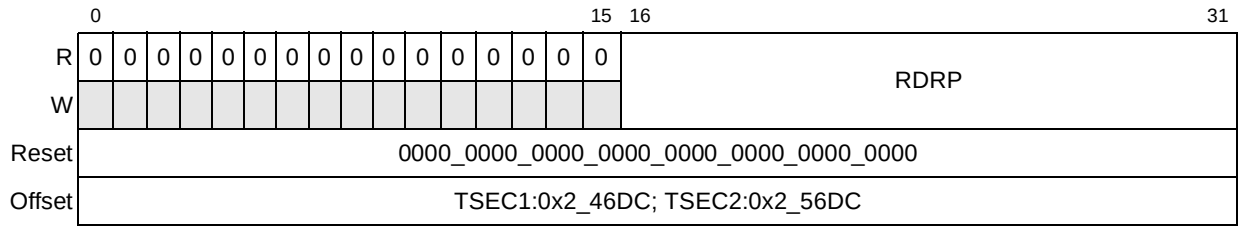


Figure 14-72. Receive Dropped Packet Counter Register Definition

Table 14-70 describes the fields of the RDRP register.

Table 14-70. 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.

14.5.3.7.25 Transmit Byte Counter Register (TBYT)

Figure 14-73 shows the TBYT register.

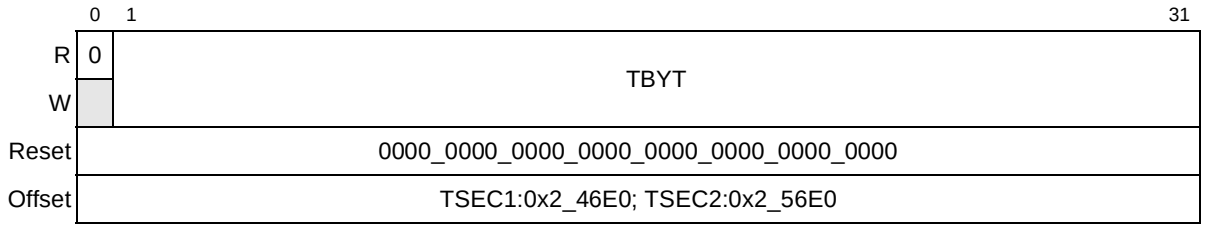


Figure 14-73. Transmit Byte Counter Register Definition

Table 14-71 describes the fields of the TBYT register.

Table 14-71. 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.

14.5.3.7.26 Transmit Packet Counter Register (TPKT)

Figure 14-74 shows the TPKT register.

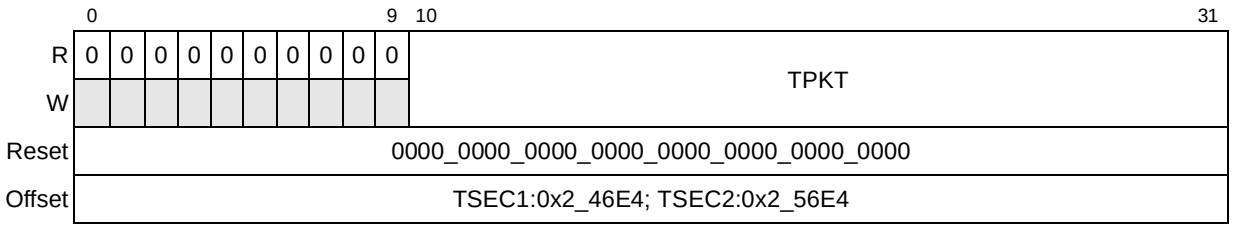


Figure 14-74. Transmit Packet Counter Register Definition

Table 14-72 describes the fields of the TPKT register.

Table 14-72. 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).

14.5.3.7.27 Transmit Multicast Packet Counter Register (TMCA)

Figure 14-75 shows the TMCA register.

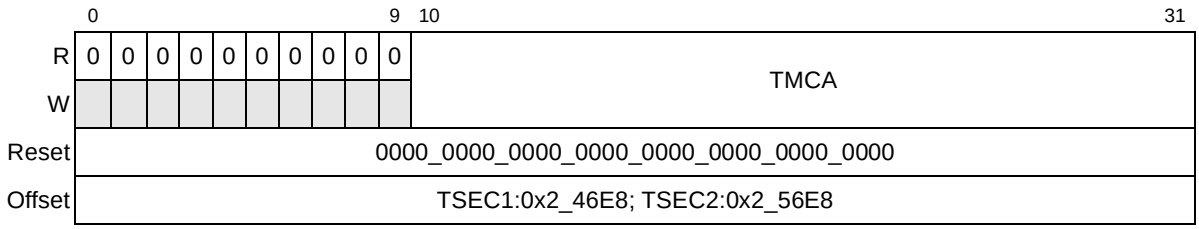


Figure 14-75. Transmit Multicast Packet Counter Register Definition

Table 14-73 describes the fields of the TMCA register.

Table 14-73. 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).

14.5.3.7.28 Transmit Broadcast Packet Counter Register (TBCA)

Figure 14-76 shows the TBCA register.

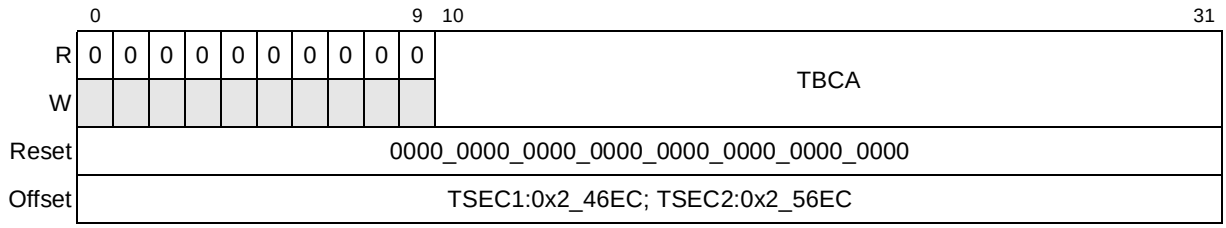


Figure 14-76. Transmit Broadcast Packet Counter Register Definition

Table 14-74 describes the fields of the TBCA register.

Table 14-74. 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).

14.5.3.7.29 Transmit Pause Control Frame Counter Register (TXPF)

Figure 14-77 shows the TXPF register.

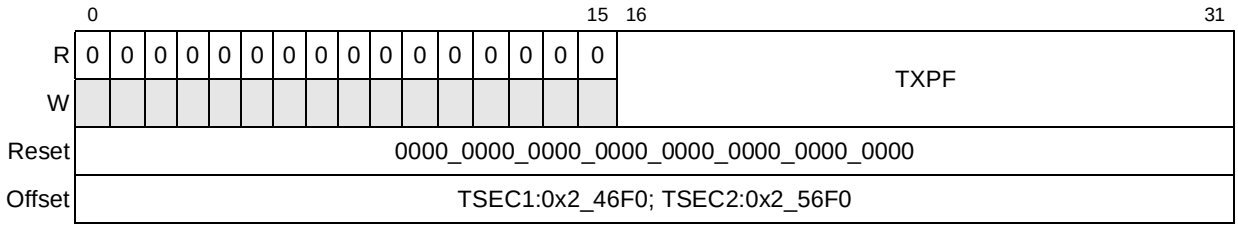


Figure 14-77. Transmit Pause Control Frame Counter Register Definition

Table 14-75 describes the fields of the TXPF register.

Table 14-75. 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.

14.5.3.7.30 Transmit Deferral Packet Counter Register (TDFR)

Figure 14-78 shows the TDFR register.

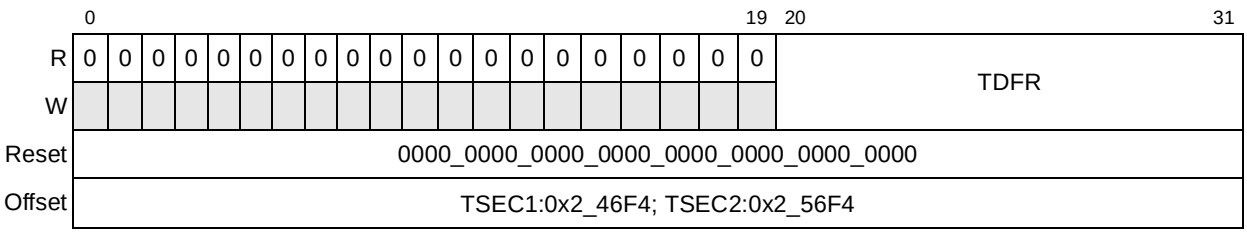


Figure 14-78. Transmit Deferral Packet Counter Register Definition

Table 14-76 describes the fields of the TDFR register.

Table 14-76. 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.

14.5.3.7.31 Transmit Excessive Deferral Packet Counter Register (TEDF)

Figure 14-79 shows the TEDF register.

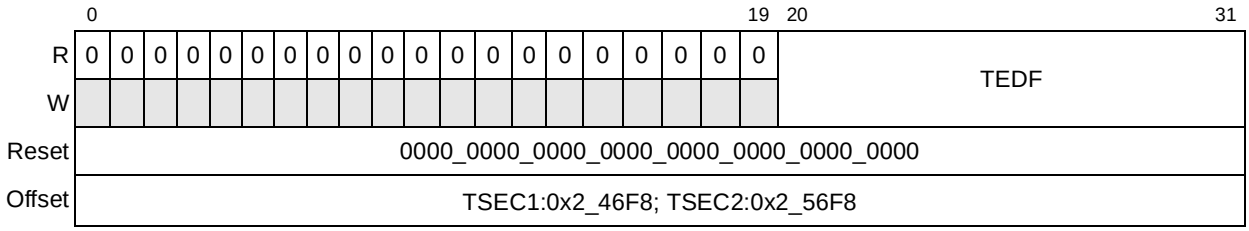


Figure 14-79. Transmit Excessive Deferral Packet Counter Register Definition

Table 14-77 describes the fields of the TEDF register.

Table 14-77. 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).

14.5.3.7.32 Transmit Single Collision Packet Counter Register (TSCL)

Figure 14-80 shows the TSCL register.

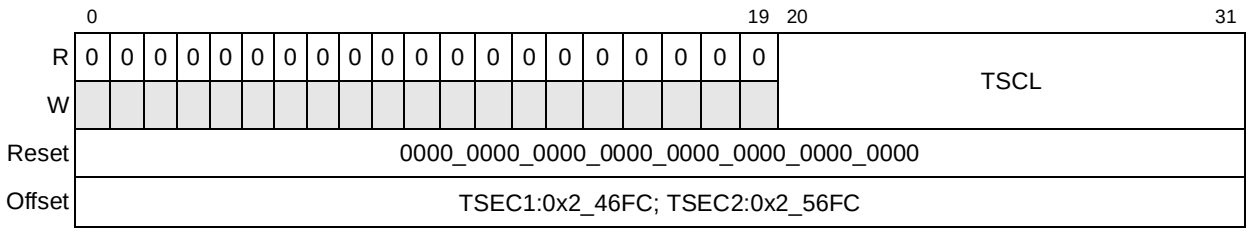


Figure 14-80. Transmit Single Collision Packet Counter Register Definition

Table 14-78 describes the fields of the TSCL register.

Table 14-78. 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.

14.5.3.7.33 Transmit Multiple Collision Packet Counter Register (TMCL)

Figure 14-81 shows the TMCL register.

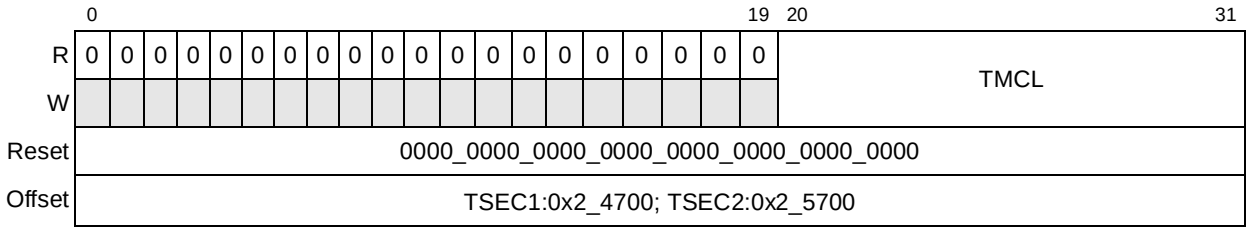


Figure 14-81. Transmit Multiple Collision Packet Counter Register Definition

Table 14-79 describes the fields of the TMCL register.

Table 14-79. 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.

14.5.3.7.34 Transmit Late Collision Packet Counter Register (TLCL)

Figure 14-82 shows the TLCL register.

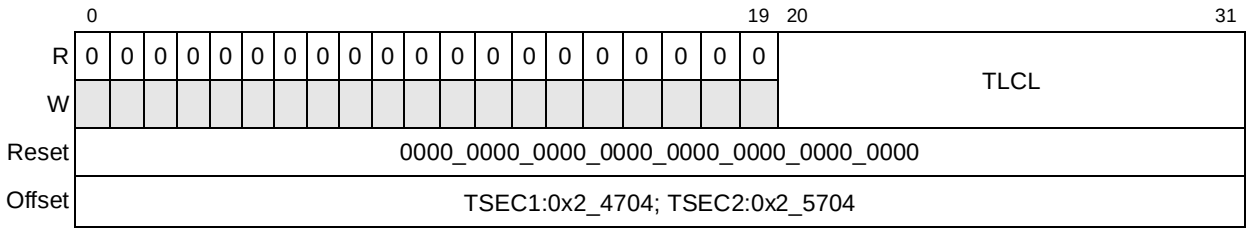


Figure 14-82. Transmit Late Collision Packet Counter Register Definition

Table 14-80 describes the fields of the TLCL register.

Table 14-80. 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.

14.5.3.7.35 Transmit Excessive Collision Packet Counter Register (TXCL)

Figure 14-83 shows the TXCL register.

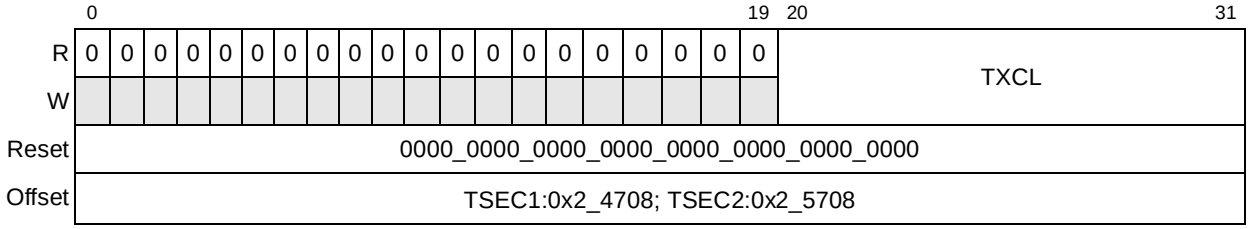


Figure 14-83. Transmit Excessive Collision Packet Counter Register Definition

Table 14-81. 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.

14.5.3.7.36 Transmit Total Collision Counter Register (TNCL)

Figure 14-84 shows the TNCL register.

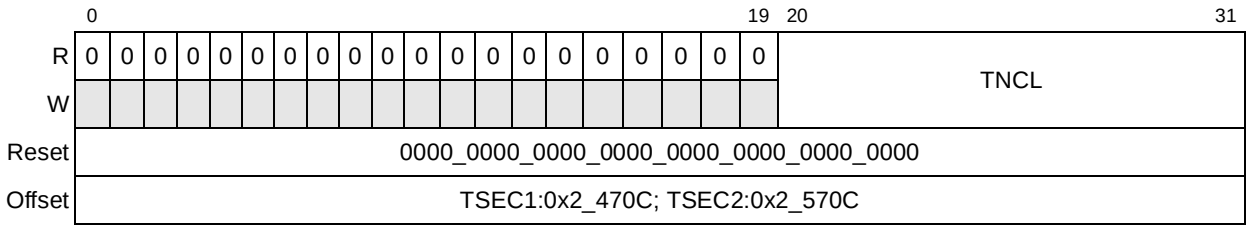


Figure 14-84. Transmit Total Collision Counter Register Definition

Table 14-82 describes the fields of the TNCL register.

Table 14-82. 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.

14.5.3.7.37 Transmit Drop Frame Counter Register (TDRP)

Figure 14-85 shows the TDRP register.

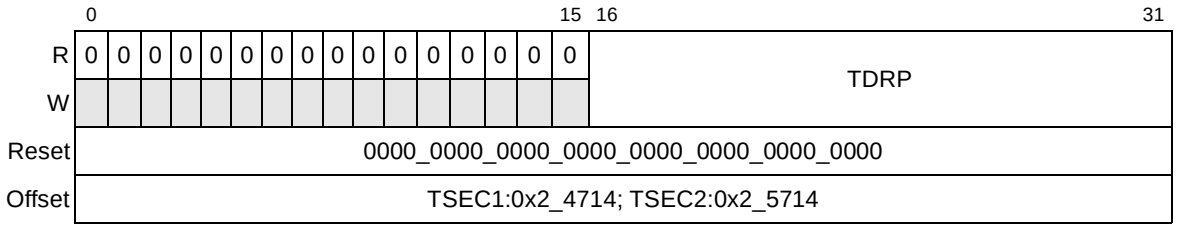


Figure 14-85. Transmit Drop Frame Counter Register Definition

Table 14-83 describes the fields of the TDRP register.

Table 14-83. 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.

14.5.3.7.38 Transmit Jabber Frame Counter Register (TJBR)

Figure 14-86 shows the TJBR register.

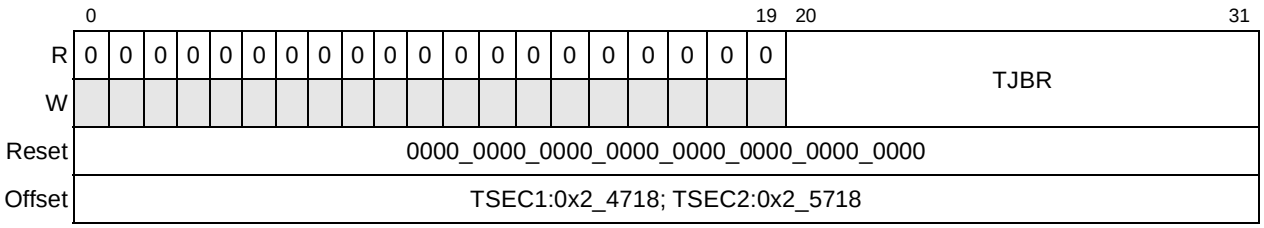


Figure 14-86. Transmit Jabber Frame Counter Register Definition

Table 14-84 describes the fields of the TJBR register.

Table 14-84. 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.

14.5.3.7.39 Transmit FCS Error Counter Register (TFCS)

Figure 14-87 shows the TFCS register.

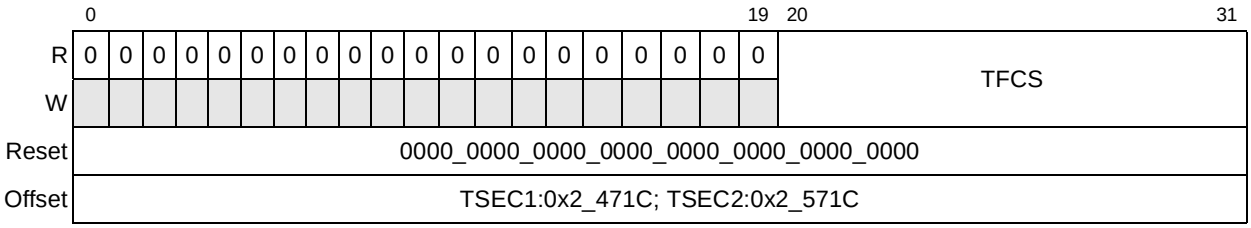


Figure 14-87. Transmit FCS Error Counter Register Definition

Table 14-85 describes the fields of the TFCS register.

Table 14-85. 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.

14.5.3.7.40 Transmit Control Frame Counter Register (TXCF)

Figure 14-88 shows the TXCF register.

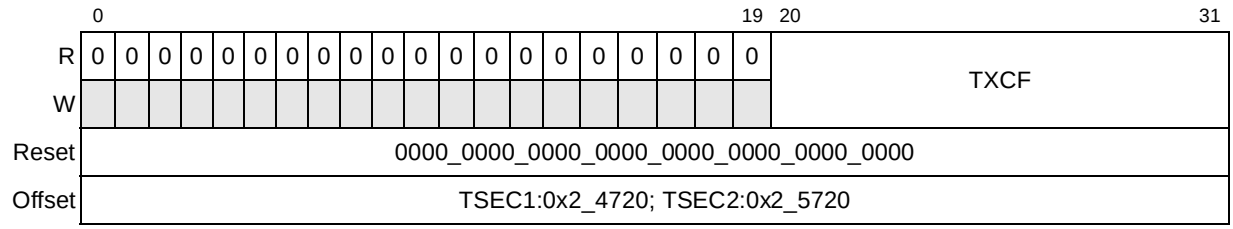


Figure 14-88. Transmit Control Frame Counter Register Definition

Table 14-86 describes the fields of the TXCF register.

Table 14-86. 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.

14.5.3.7.41 Transmit Oversize Frame Counter Register (TOVR)

Figure 14-89 shows the TOVR register.

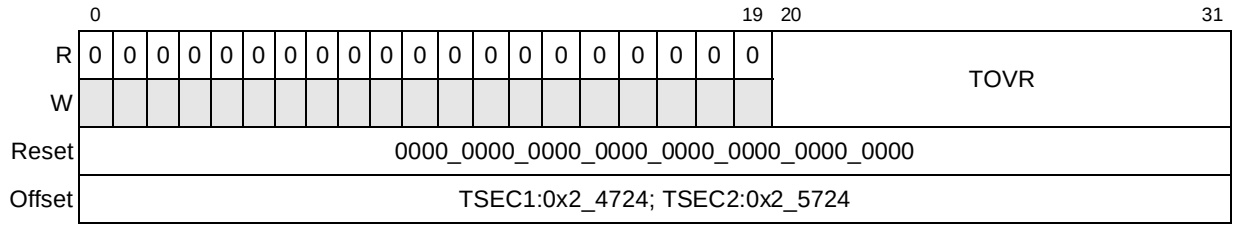


Figure 14-89. Transmit Oversized Frame Counter Register Definition

Table 14-87 describes the fields of the TOVR register.

Table 14-87. 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.

14.5.3.7.42 Transmit Undersize Frame Counter Register (TUND)

Figure 14-90 shows the TUND register.

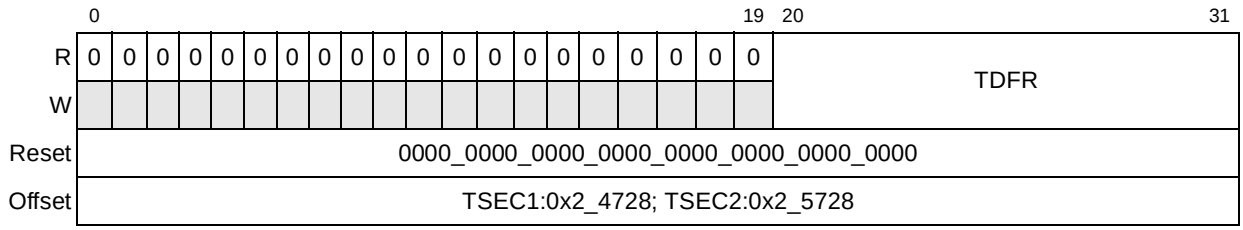


Figure 14-90. Transmit Undersize Frame Counter Register Definition

Table 14-88 describes the fields of the TUND register.

Table 14-88. 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.

14.5.3.7.43 Transmit Fragment Counter Register (TFRG)

Figure 14-91 shows the TFRG register.

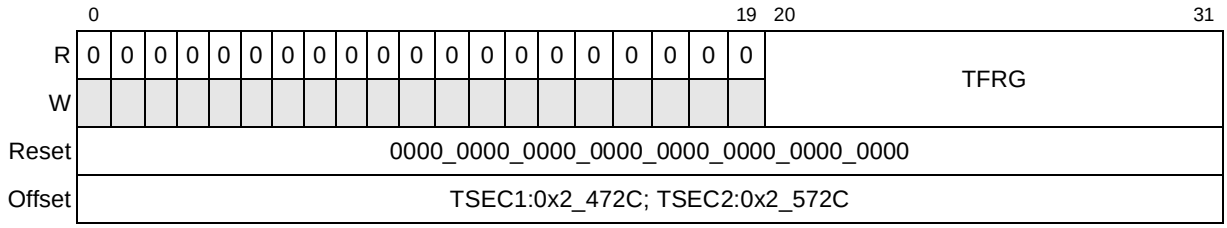


Figure 14-91. Transmit Fragment Counter Register Definition

Table 14-89 describes the fields of the TFRG register.

Table 14-89. 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.

14.5.3.7.44 Carry Register 1 (CAR1)

Carry register bits are cleared when written with a one. Figure 14-92 shows the CAR1 register.

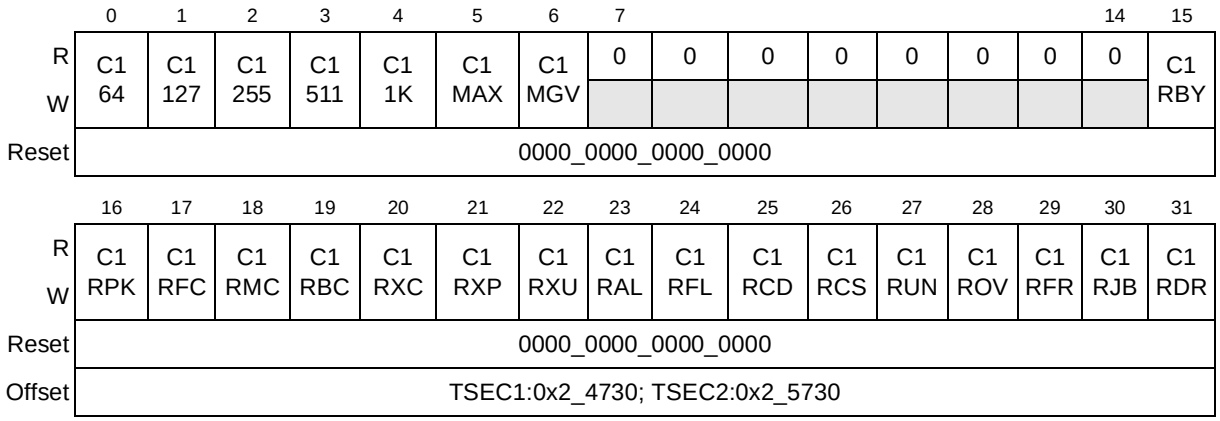


Figure 14-92. Carry Register 1 (CAR1) Register Definition

Table 14-90 describes the fields of the CAR1 register.

Table 14-90. 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

14.5.3.7.45 Carry Register 2 (CAR2)

Carry register bits are cleared when written with a one. Figure 14-93 shows the CAR2 register.

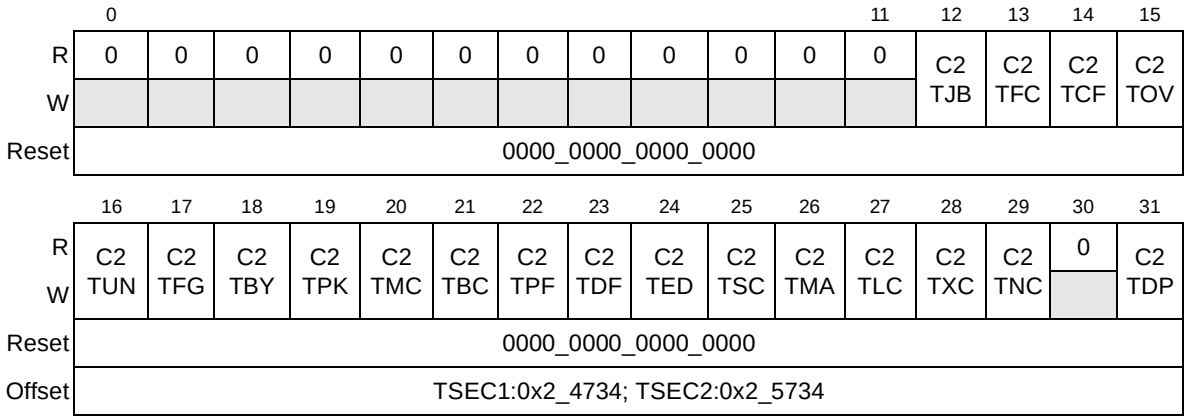


Figure 14-93. Carry Register 2 (CAR2) Register Definition

Table 14-91 describes the fields of the CAR2 register.

Table 14-91. 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 14-91. 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

14.5.3.7.46 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 14-94 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 14-94. Carry Mask Register 1 (CAM1) Register Definition

Table 14-92 describes the fields of the CAM1 register.

Table 14-92. 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 14-92. 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

14.5.3.7.47 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 14-95 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 14-95. Carry Mask Register 2 (CAM2) Register Definition

Table 14-93 describes the fields of the CAM2 register.

Table 14-93. 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

14.5.3.8 Hash Function Registers

This section provides detailed descriptions of the registers used for hash functions. All of the registers are 32 bits wide. If the DA field of a receive frame is processed through a 32-bit CRC generator, the 8 bits of the CRC remainder is mapped to a hash table entry. The user can enable a hash entry by setting the appropriate bit. A hash entry usually represents a set of addresses. A hash table hit occurs if the DA CRC result points to an enabled hash entry. The user must further filter the address. See Section 14.6.2.6.2, “Hash Table Algorithm,” for more information on the hash algorithm.

14.5.3.8.1 Individual Address Registers 0–7 (IADDR*n*)

The IADDR*n* registers, shown in Figure 14-96, 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.

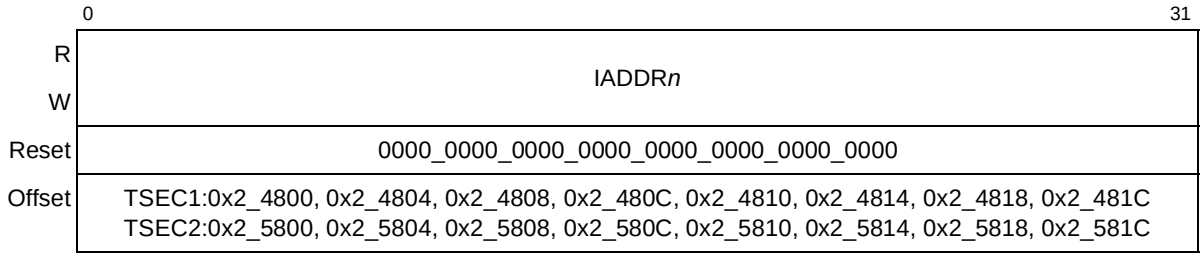


Figure 14-96. IADDR*n* Register Definition

Table 14-94 describes the field of the IADDR*n* registers.

Table 14-94. 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.

14.5.3.8.2 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 14-97 shows the GADDR*n* registers.

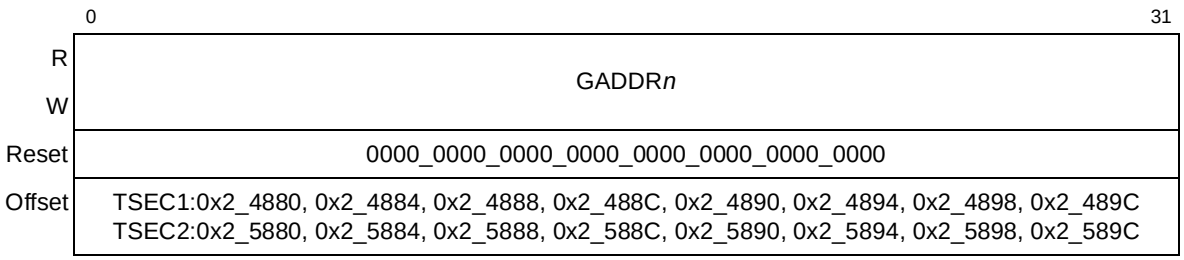


Figure 14-97. GADDR*n* Register Definition

Table 14-95 describes GADDR*n*.

Table 14-95. 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.

14.5.3.9 Attribute Registers

This section describes the two TSEC frame attribute registers.

14.5.3.9.1 Attribute Register (ATTR)

The attribute register, shown in Figure 14-98, 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.

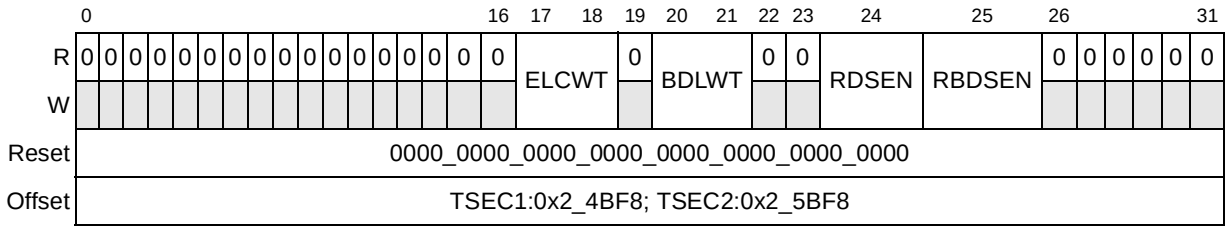


Figure 14-98. ATTR Register Definition

Table 14-96 describes the fields of the ATTR register.

Table 14-96. 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 14-96. 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

14.5.3.9.2 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 14-99 shows the ATTRELI register.

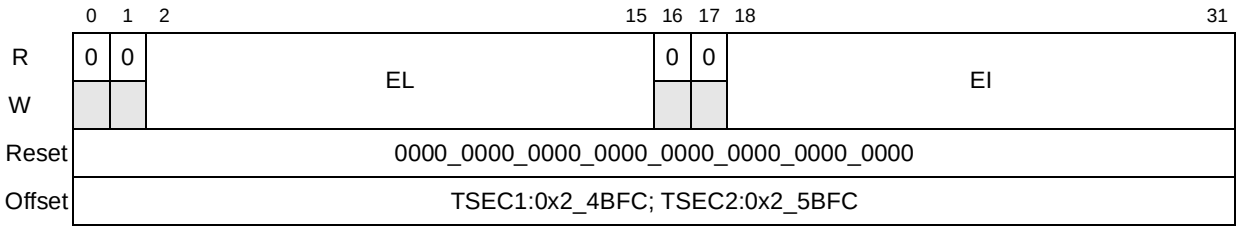


Figure 14-99. ATTRELI Register Definition

Table 14-97 describes the fields of the ATTRELI register.

Table 14-97. 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.

14.5.4 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) and the TBI MII set of registers.

14.5.4.1 TBI MII Set Register Descriptions

This section describes the TBI MII registers. All 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 14-98), 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 14.5.3.1, "TSEC General Control and Status Registers," and the TBI MII register set in Table 14-98.

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 14-98. TBI MII Register Set

Offset	Name	Access	Size	Section/Page
TEN-BIT INTERFACE (TBI) REGISTERS				14.5.4/14-91
0x00	Control (CR)	R/W[1]	16 bits	14.5.4.2/14-92
0x01	Status (SR)	R, LH, LL	16 bits	14.5.4.3/14-93
0x02–0x03	Reserved	R	2 bytes	—
0x04	AN advertisement (ANA)	RW, R	16 bits	14.5.4.3/14-93
0x05	AN link partner base page ability (ANLPBPA)	R	16 bits	14.5.4.5/14-96
0x06	AN expansion (ANEX)	R, LH	16 bits	14.5.4.6/14-98
0x07	AN next page transmit (ANNPT)	R/W, R	16 bits	14.5.4.7/14-98
0x08	AN link partner ability next page (ANLPANP)	R	16 bits	14.5.4.8/14-99
0x0F	Extended status (EXST)	R	16 bits	14.5.4.9/14-100
0x10	Jitter diagnostics (JD)	R/W	16 bits	14.5.4.10/14-101
0x11	TBI control (TBICON)	R/W	16 bits	14.5.4.11/14-102

[1] R = means read only, WO = write only, R/W = read and write, LH = latches high, LL = latches low, SC = self-clearing.

14.5.4.2 Control Register (CR)

Figure 14-100 shows the CR register.

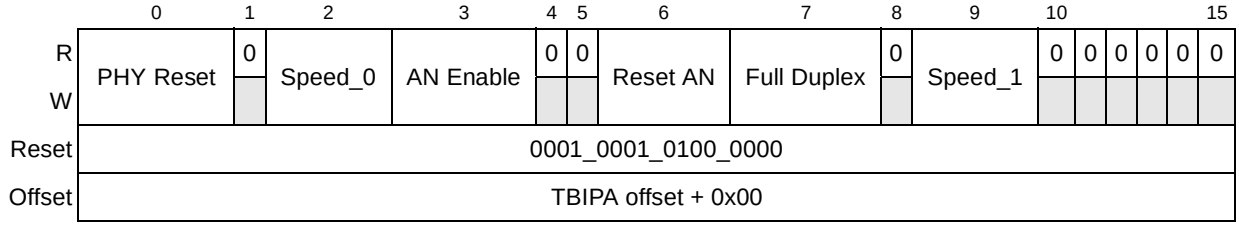


Figure 14-100. Control Register Definition

Table 14-99 describes the fields of the CR register.

Table 14-99. 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 14-99. 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

14.5.4.3 Status Register (SR)

Figure 14-101 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 14-101. Status Register Definition

Table 14-100 describes the fields of the SR register.

Table 14-100. 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 14-100. 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.

14.5.4.4 AN Advertisement Register (ANA)

Figure 14-102 shows the ANA register.

	0	1	2 3	4		6	7 8	9	10	11				15
R	Next Page	Ack	Remote Fault	0	0	0	Pause	Half Duplex	Full Duplex	0	0	0	0	0
W														
Reset	0000_0000_0000_0000													
Offset	TBIPA offset + 0x04													

Figure 14-102. AN Advertisement Register Definition

Table 14-101 describes the fields of the ANA register.

Table 14-101. 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 14-101. 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 14-102.
<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 14-102 describes the resolution of pause priority.

Table 14-102. 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

14.5.4.5 AN Link Partner Base Page Ability Register (ANLPBPA)

Figure 14-103 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 14-103. AN Link Partner Base Page Ability Register Definition

Table 14-103 describes the fields of the ANLPBPA register.

Table 14-103. 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>

14.5.4.6 AN Expansion Register (ANEX)

Figure 14-104 shows the ANEX register.

	0												12	13	14	15
R	0	0	0	0	0	0	0	0	0	0	0	0	0	NP Able	Page Rx'd	0
W																
Reset	0000_0000_0000_0110															
Offset	TBIPA offset + 0x06															

Figure 14-104. AN Expansion Register Definition

Table 14-104 describes the fields of the ANEX register.

Table 14-104. 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.

14.5.4.7 AN Next Page Transmit Register (ANNPT)

Figure 14-105 shows the ANNPT 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 + 0x07						

Figure 14-105. AN Next Page Transmit Register Definition

Table 14-105 describes the fields of the ANNPT register.

Table 14-105. 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]

14.5.4.8 AN Link Partner Ability Next Page Register (ANLPANP)

Figure 14-106 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 14-106. AN Link Partner Ability Next Page Register Definition

Table 14-106 describes the ANLPANP fields.

Table 14-106. 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 14-106. 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.

14.5.4.9 Extended Status Register (EXST)

Figure 14-107 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 14-107. Extended Status Register Definition

Table 14-107 describes the fields of the EXST register.

Table 14-107. 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 14-107. 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

14.5.4.10 Jitter Diagnostics Register (JD)

Annex 36A in IEEE 802.3z describes several jitter test patterns. These can be configured to be sent by writing the jitter diagnostics register. See the register description for more information. It may be wise to auto-negotiate and advertise a remote fault, signaling offline, prior to beginning the test patterns. Figure 14-108 shows the JD register.

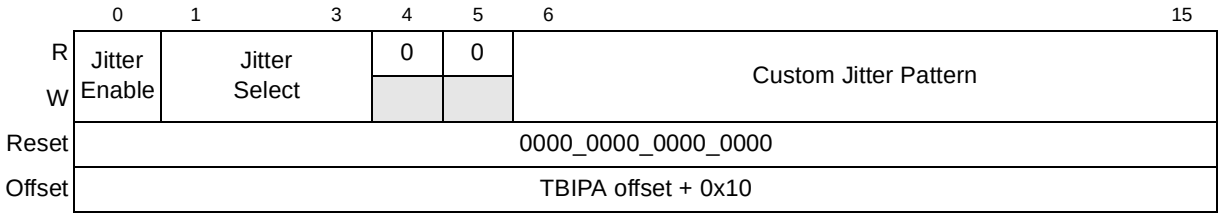


Figure 14-108. Jitter Diagnostics Register Definition

Table 14-108 describes the fields of the JD register.

Table 14-108. JD Field Description

Bits	Name	Description
0	Jitter Enable	Jitter enable. This bit is cleared by default. 0 Normal transmit operation 1 Enable the TBI to transmit the jitter test patterns defined in annex 36A of the *IEEE 802.3 specification*.
1–3	Jitter Select	Selects the jitter pattern to be transmitted in diagnostics mode. Encoding of this field is shown in the following table. Default is 0

Jitter Pattern Select	bit[1]	bit[2]	bit[3]
User defined uses custom jitter pattern, bits 6–15	0	0	0
High frequency (±D21.5) 1010101010101010101010101010101010101010…	0	0	1
Mixed frequency (±K28.5) 1111101011000001010011111010110000010100…	0	1	0
Low frequency 1111100000111110000011111000001111100000…	0	1	1
Complex pattern (10’h17c,10’h0c9,10’h0e5,10’h2a3, 10’h17c,...)	1	0	0
Square Wave (-K28.7) 0011111000001111100000111110000011111000…	1	0	1
Reserved	1	1	0

	0	1	2	3	4		6	7	8	9	10	11	12	13	14	15
R	Soft_Reset	0	Disable Rx Dis	Disable Tx Dis	0	0	0	AN Sense	0	0	Clock Select	MII Mode	0	0	0	0
W																
Reset	0000_0000_0000_0000															
Offset	TBIPA offset + 0x11															

Figure 14-109. TBI Control Register Definition

14.5.4.11 TBI Control Register (TBICON)

Figure 14-109 shows the TBICON register.

Table 14-109 describes the fields of the TBICON register.

Table 14-109. 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

14.6 Functional Description

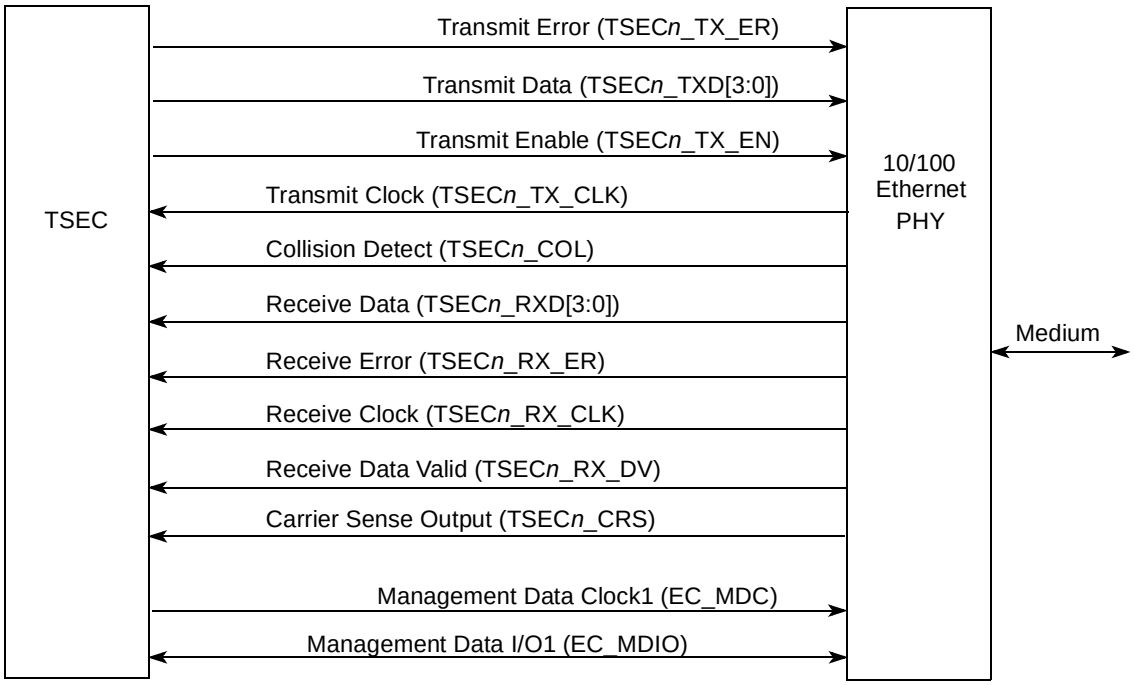
This section describes many of the functions of the TSEC controller.

14.6.1 Connecting to Physical Interfaces

This section describes how to connect the TSEC to various interfaces: MII, GMII, RGMII, RTBI and TBI. To avoid confusion, all of the buses follow the bus conventions used in the IEEE 802.3 specification, because 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).

14.6.1.1 Media-Independent Interface (MII)

This section describes the media-independent interface (MII) intended to be used between the PHYs and the TSEC. Figure 14-110 shows the basic components of the MII including the signals required to establish TSEC module connection with a PHY.



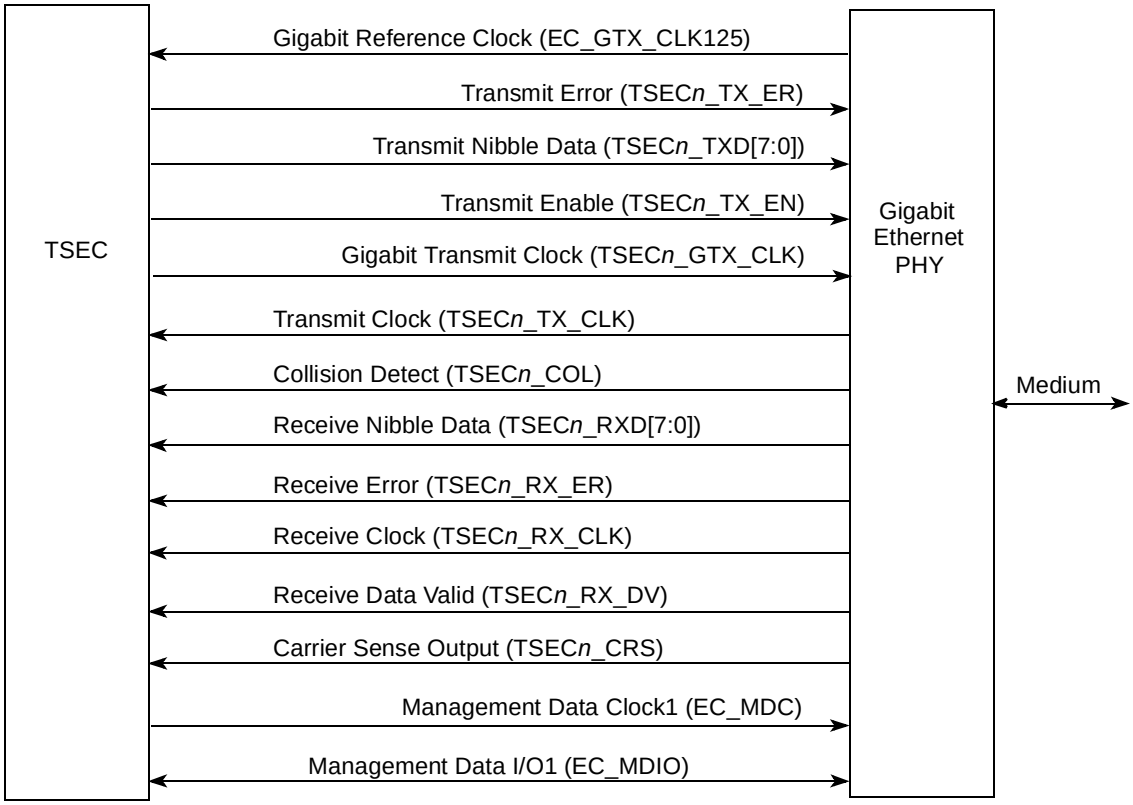
[1] The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 14-110. 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.

14.6.1.2 Gigabit Media-Independent Interface (GMII)

This section describes the gigabit media-independent interface (GMII) intended to be used between the PHYs and the TSEC. Figure 14-111 shows the basic components of the GMII including the signals required to establish the TSEC module connection with a PHY.



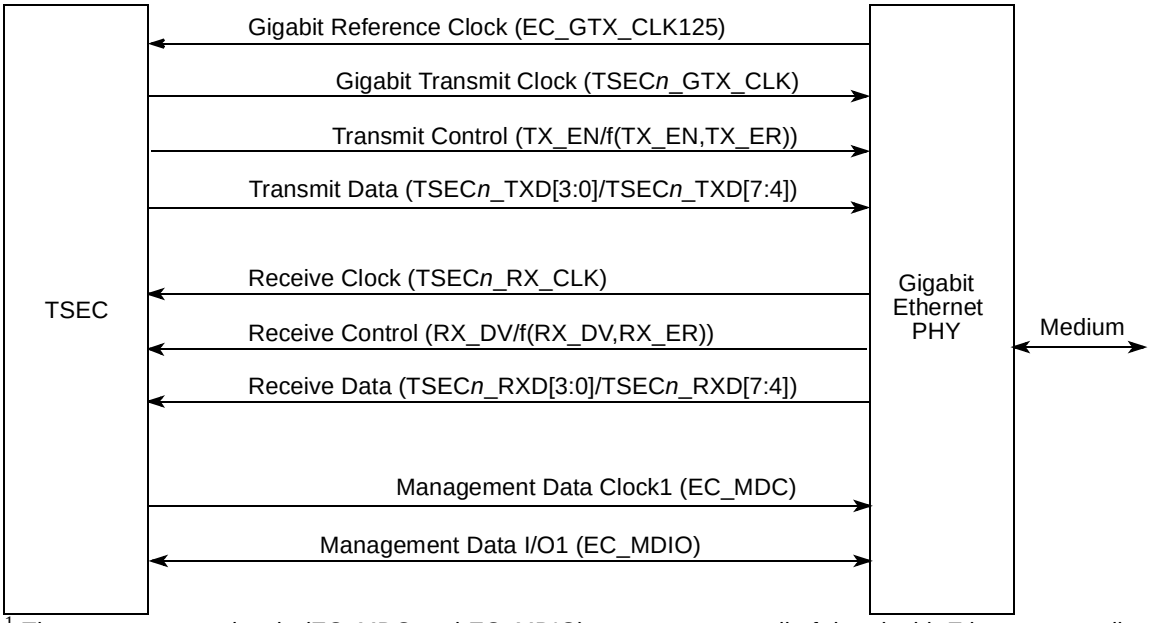
[1] The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 14-111. 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.

14.6.1.3 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 14-112 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.

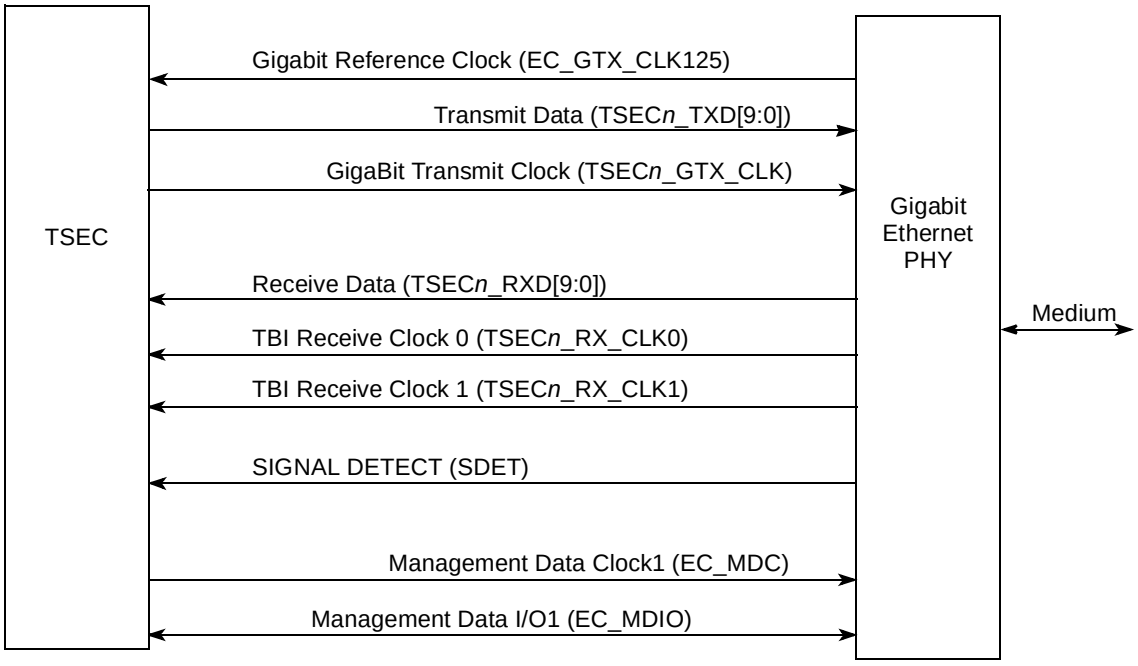


[1] The management signals (EC_MDC and EC_MDIO) are common to all of the gigabit Ethernet controllers module connections in the system, assuming that each PHY has a different management address.

Figure 14-112. TSEC-RGMII Connection

14.6.1.4 Ten-Bit Interface (TBI)

This section describes the ten-bit interface (TBI) intended to be used between the PHYs and the TSEC to implement a standard SerDes interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 14-113 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.



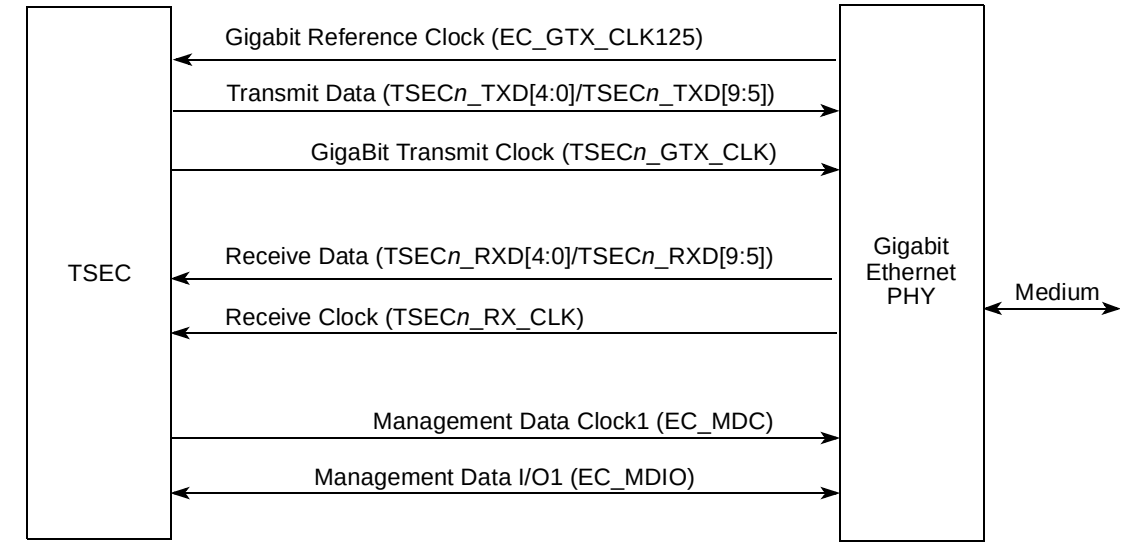
[1] The management signals (EC_MDC and EC_MDIO) are common to all of the Ethernet controllers' connections in the system, assuming that each PHY has a different management address.

Figure 14-113. 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.

14.6.1.5 Reduced Ten-Bit Interface (RTBI)

This section describes the reduced ten-bit interface (RTBI) intended to be used between the PHYs and the TSEC to implement a reduced signal count version of a SerDes interface for optical-fiber devices in 1000BASE-SX/LX applications. Figure 14-114 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 14-114. 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 14-110 describes the signal multiplexing for GMII, MII and TBI interfaces.

Table 14-110. 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 14-110. 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 14-111 describes the signal multiplexing for RGMII and RTBI interfaces.

Table 14-111. 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 14-111. 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 14-112 describes the signals shared by all interfaces.

Table 14-112. 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	

14.6.2 Gigabit Ethernet Channel Operation

This section describes the operation of the TSEC. First, the software initialization sequence is described. Next, the software (Ethernet driver) interface for transmitting and receiving frames is reviewed. Address recognition and hash table algorithm features are also discussed. The section concludes with interrupt handling, inter-packet gap time, and loop back descriptions.

14.6.2.1 Initialization Sequence

This 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.

14.6.2.1.1 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.

14.6.2.1.2 User Initialization

After the system has undergone a hard reset, software must initialize certain basic TSEC registers. Other registers can also be initialized during this time, but they are optional and must be determined based on the requirements of the system. The module memory map in Table 14-3 lists all the TSEC registers. Table 14-113 describes the minimum steps for register initialization.

Table 14-113. Steps of Minimum Register Initialization

Description
1. Set, then clear MACCFG1 [Soft_Reset]
2. Initialize MACCFG2
3. Initialize MAC station address

Table 14-113. 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 14.5.3.1.7, “DMA Control Register (DMACTRL),” and Section 14.6.3.1, “Transmit Data Buffer Descriptor (TxBD),” for more information.

14.6.2.2 Soft Reset and Reconfiguring Procedure

Before issuing a soft reset to and/or reconfiguring the MAC with new parameters, user must properly shutdown the DMA and make sure it is in an idle state for the entire duration. User must gracefully stop the DMA by setting both GRS and GTS bits in the DMACTRL register, then wait for both GRSC and GTSC bits to be set in the IEVENT register before resetting the MAC or changing parameters. Both GRS and GTS bits must be cleared before re-enabling the MAC to resume the DMA.

During the MAC configuration, if a new set of Tx buffer descriptors 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.

14.6.2.3 Gigabit Ethernet Frame Transmission

The Ethernet transmitter requires little core intervention. After the software driver initializes the system, the TSEC begins to poll the first transmit buffer descriptor (TxBD) in the TxBD ring every 512 transmit clocks. If TxBD[R] is set, TSEC begins moving transmit buffer from memory to its Tx FIFO. The transmitter takes data from the Tx FIFO and transmits data to the MAC. The MAC transmits the data through the GMII interface to the physical media. The transmitter, once initialized, runs until the end-of-frame (EOF) condition is detected unless a collision within the collision window occurs (half-duplex mode) or an abort condition is encountered.

If the user has a frame ready to transmit, 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.

14.6.2.4 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 14.6.4, “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 14.6.2.6, “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.

14.6.2.5 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.

14.6.2.6 Frame Recognition

The Ethernet controller performs frame recognition using destination address (DA) recognition. A frame can be rejected or accepted based on the outcome.

14.6.2.6.1 Destination Address Recognition

The Ethernet controller can also perform the frame filtering using the traditional destination address (DA) recognition methods.

Figure 14-115 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.

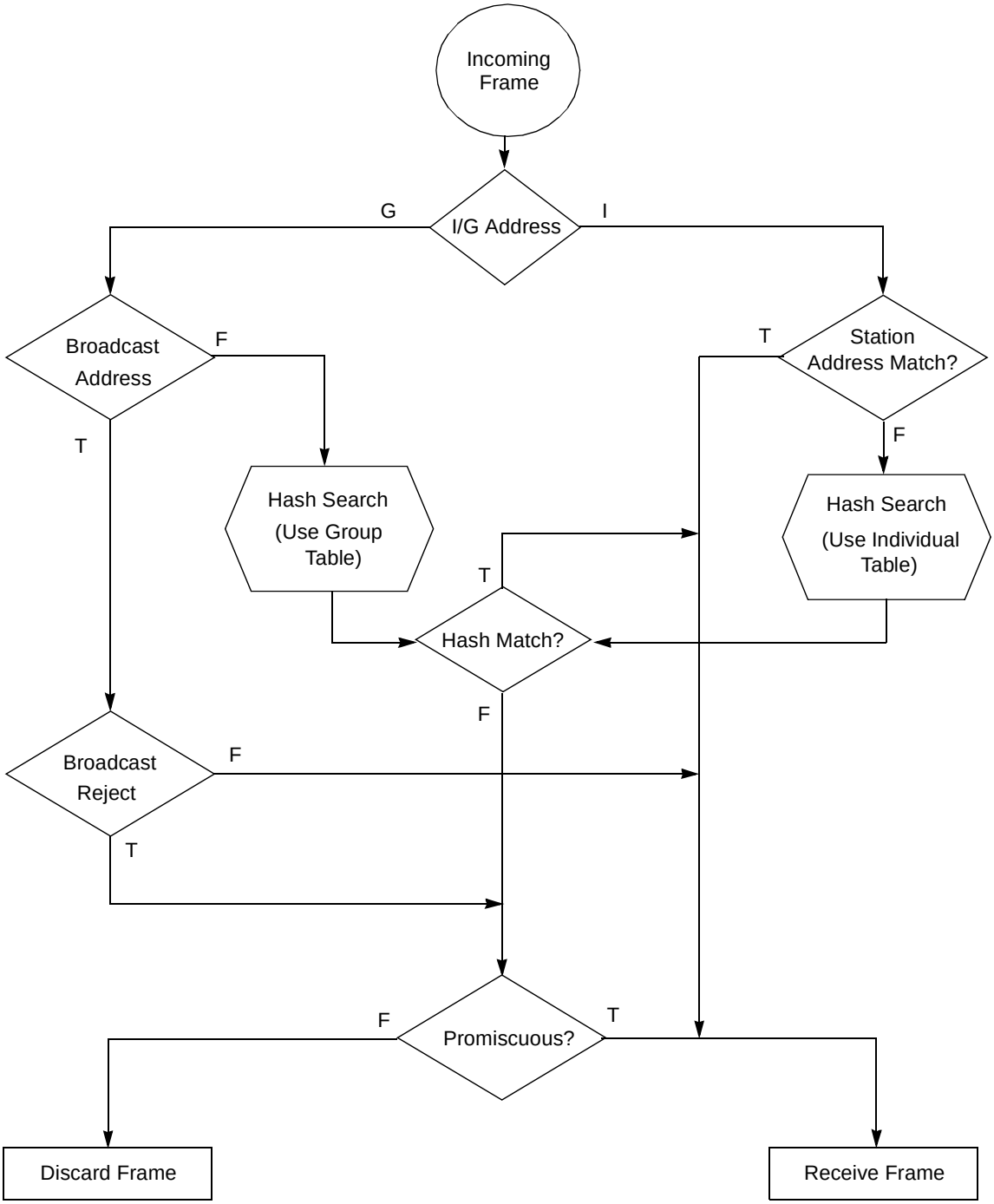


Figure 14-115. Ethernet Address Recognition Flowchart

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 14.5.3.4.1, “Receive Control Register (RCTRL).”)

In the group type of address recognition, the Ethernet controller determines whether the group address is a broadcast address. If it is a broadcast, and broadcast addresses are enabled, the frame is accepted. If the group address is not a broadcast address, the user can perform address recognition on multiple group addresses using the GADDR*n* hash table. In promiscuous mode, the Ethernet controller receives all of the incoming frames regardless of their address.

14.6.2.6.2 Hash Table Algorithm

The hash table process used in the individual and group hash filtering operates as follows. The Ethernet controller maps any 48-bit destination address into one of 256 bins, represented by the 256 bits in GADDR0–7 or IADDR0–7. The eight high-order bits of a cyclic redundancy check (CRC) checksum are used to index into the hash table. The high-order three bits of this 8-bit field are used to select one of the eight registers in either the individual or group hash table. The low-order five bits select a bit within the 32-bit register. A value of 0 in the high-order three bits selects IADDR0/GADDR0.

The same process is used if the Ethernet controller receives a frame. If the CRC checksum selects a bit that is set in the group/individual hash table, the frame is accepted. If 32 group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 224/256 (87.5%) of the group address frames from reaching memory. Software must further filter those that reach memory to determine if they contain the correct addresses.

Better performance is achieved by using the group and individual hash tables in combination. For instance, if 32 group and 32 physical addresses are stored in their respective hash tables, because 87.5% of all group addresses and 87.5% of all individual address are rejected, then 87.5% of all frames are prevented from reaching memory.

The effectiveness of the hash table declines as the number of addresses increases. For instance, as the number of addresses stored in the 256-bin hash table increases, the vast majority of the hash table bits are set, preventing only a small fraction of frames from reaching memory.

14.6.2.6.3 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

14.6.2.7 Flow Control

Because collisions cannot occur in full-duplex mode, gigabit Ethernet can operate at the maximum rate. If the rate becomes too fast for a station’s receiver, the station’s transmitter can send flow-control frames to reduce the rate. Flow-control instructions are transferred by special frames of minimum frame size. The length/type fields of these frames have a special value. Table 14-114 lists the flow-control frame structure.

Table 14-114. 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.

14.6.2.8 Interrupt Handling

The following describes what usually occurs within a TSEC interrupt handler:

- If an interrupt occurs, read IEVENT to determine interrupt sources. IEVENT bits to be handled in this interrupt handler are normally cleared at this time.
- Process the TxBDs to reuse them if the IEVENT[TXB or 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 14-115.
- 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 14-116.

- Clear any set halt bits in TSTAT and RSTAT registers, or DMACTRL[GTS] and DMACTRL[GRS].
- Continue normal execution.

Table 14-115. 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 14-116. 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].

14.6.2.8.1 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.

14.6.2.8.2 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.

14.6.2.8.3 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 14-117 specifies the range of possible timing thresholds subject to the interface frequency and the value of RXIC[ICTT] or TXIC[ICTT].

Table 14-117. 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.

14.6.2.9 Inter-Packet Gap Time

If a station must transmit, it waits until the LAN becomes silent for a specified period (inter-packet gap). After a station begins sending, it continually checks for collisions on the LAN. If a collision is detected, the station forces a jam signal (all ones) on its frame and stops transmitting. Collisions usually occur close to the beginning of a packet. The station then waits a random time period (back-off) before attempting to send again. After the back-off completes, the station waits for silence on the LAN and then begins retransmission on the LAN. This process is called a retry. If the packet is not successfully sent within a specified number of retries, an error is indicated.

The minimum inter-packet gap time for back-to-back transmission is 96 serial clocks. The receiver receives back-to-back packets with this minimum spacing. In addition, after waiting a required number of clocks (based on the back-off algorithm), the transmitter waits for carrier sense to be negated before retransmitting the packet. Retransmission begins 36 serial clocks after carrier sense is negated for at least 60 serial clocks.

14.6.2.10 Internal and External Loop Back

Setting MACCFG1[Loop Back] causes the MAC transmit outputs to be looped back to the MAC receive inputs. Clearing this bit results in normal operation. This bit is cleared by default.

14.6.2.11 Error-Handling Procedure

The Ethernet controller reports frame reception and transmission error conditions using the channel BDs, the error counters, and the IEVENT register.

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 14-118.

Table 14-118. 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 14-119.

Table 14-119. 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

14.6.3 Buffer Descriptors

The TSEC buffer descriptor (BD) is modeled after the MPC8260 Fast Ethernet controller BD for ease of reuse. Drawing from the MPC8260 FEC BD programming model, the TSEC descriptor base registers point to the beginning of BD rings. The 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 14-116). 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 14-117), 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.

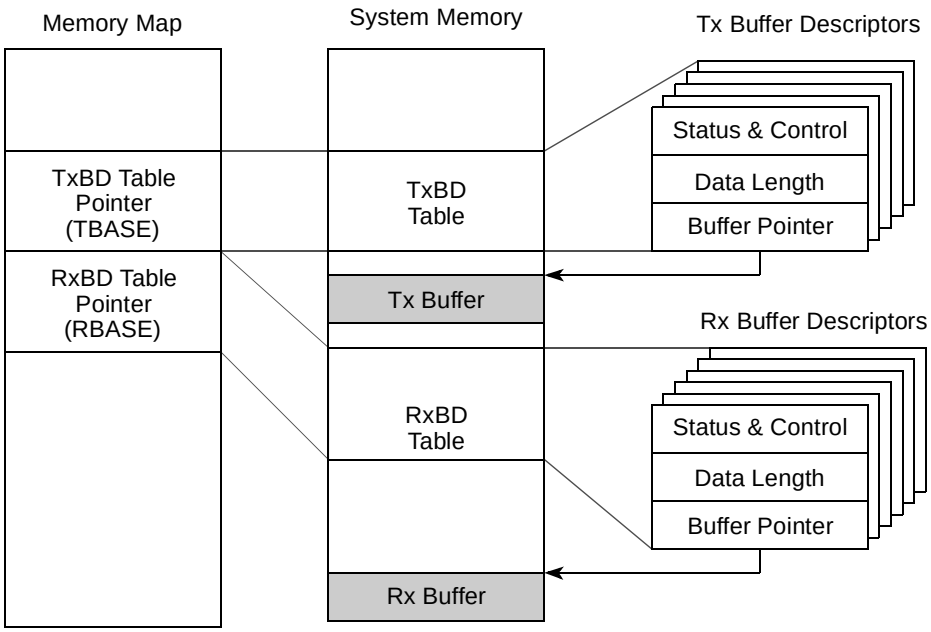


Figure 14-116. Example of TSEC Memory Structure for BD

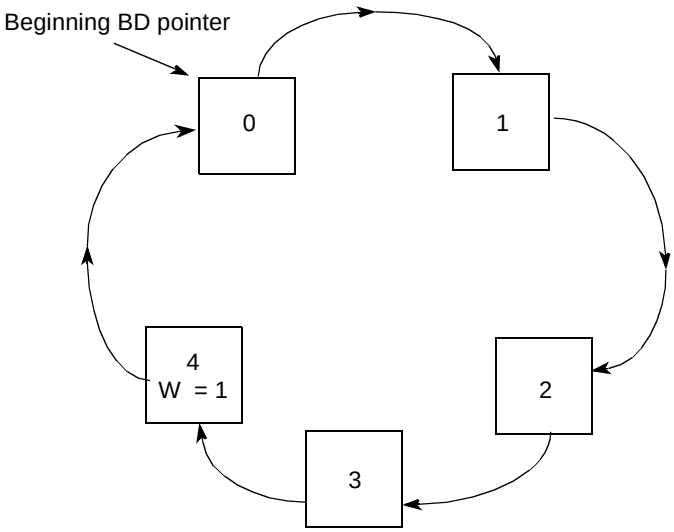


Figure 14-117. Buffer Descriptor Ring

14.6.3.1 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 14-118 shows the TxBD.

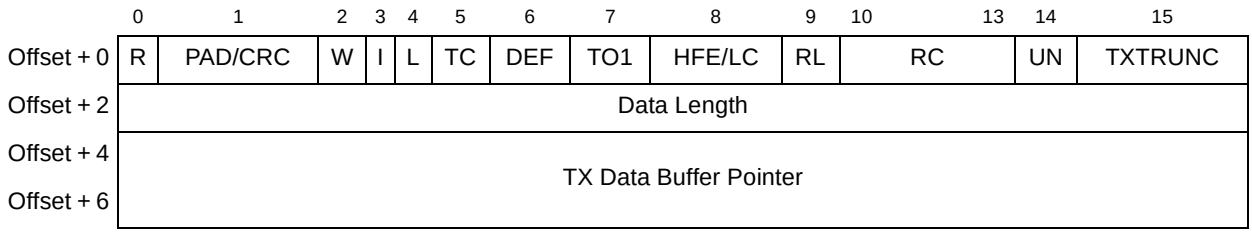


Figure 14-118. Transmit Buffer Descriptor

The TxBD fields are detailed in Table 14-120.

Table 14-120. 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 14-120. 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.

14.6.3.2 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 14-119 below.

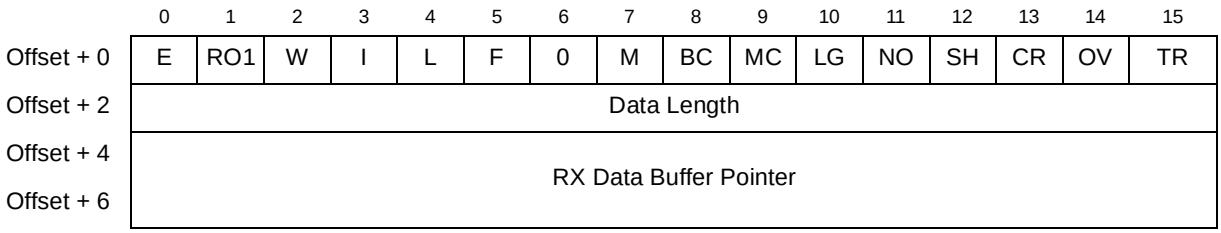


Figure 14-119. Receive Buffer Descriptor

Table 14-121 describes the fields of the RxBD.

Table 14-121. 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 14-121. 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.

14.6.4 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 14.5.3.9.1, "Attribute Register (ATTR)," and Section 14.5.3.9.2, "Attribute Extract Length and Extract Index Register (ATTRELI)," for specific register information.

14.7 Initialization/Application Information

14.7.1 Interface Mode Configuration

This section describes how to configure the TSEC in different supported interface modes. These include MII, GMII, TBI, RGMII, RTBI. The pinout, the data registers that must be initialized, as well as speed selection options are described. The ECNTRL[TBIM] and ECNTRL[RPM] bits are written, assuming the part was not pin-configured at initialization to the correct mode.

14.7.1.1 MII Interface Mode

Table 14-122 describes the signal configurations required for MII interface mode.

Table 14-122. 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 14-122. 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 14-123 describes the shared signals of the MII interface.

Table 14-123. 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 14-124 describes the register initializations required to reconfigure the PHY to MII mode following initial auto-negotiation.

Table 14-124. 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 14-124. 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 14-124. 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]

14.7.1.2 GMII Interface Mode

Table 14-125 describes the signal configurations required for GMII interface mode.

Table 14-125. 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 14-126 describes the shared signals of the GMII interface.

Table 14-126. 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 14-127 describes the register initializations required to reconfigure the PHY to GMII mode following initial auto-negotiation.

Table 14-127. 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 14-127. 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 14-127. 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]

14.7.1.3 TBI Interface Mode

Table 14-128 describes the signal configurations required for TBI interface mode.

Table 14-128. 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 14-128. 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 14-129 describes the shared signals for the TBI interface.

Table 14-129. 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 14-130 describes the register initializations required to reconfigure the PHY to TBI mode following initial auto-negotiation.

Table 14-130. 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 14-130. 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 14-130. 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]

14.7.1.4 RGMII Interface Mode

Table 14-131 shows the signals configurations required for RGMII interface mode.

Table 14-131. 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 14-131. 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 14-132 describes the shared signals for the RGMII interface.

Table 14-132. 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 14-133 describes the register initializations required to reconfigure the PHY to RGMII mode following initial auto-negotiation.

Table 14-133. 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 14-133. 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 14-133. 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]

14.7.1.5 RTBI Interface Mode

Table 14-134 describes the signal configurations required for RTBI interface mode.

Table 14-134. 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 14-134. 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 14-135 describes the shared signals for the RTBI interface.

Table 14-135. 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 14-136 describes the register initializations required to reconfigure the PHY to RTBI mode following initial auto-negotiation.

Table 14-136. 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 14-136. 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 14-136. 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 14-136. 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 15
DMA Controller

This chapter describes the DMA controller of this device.

15.1 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.

15.1.1 Block Diagram

Figure 15-1 shows the block diagram of the DMA controller.

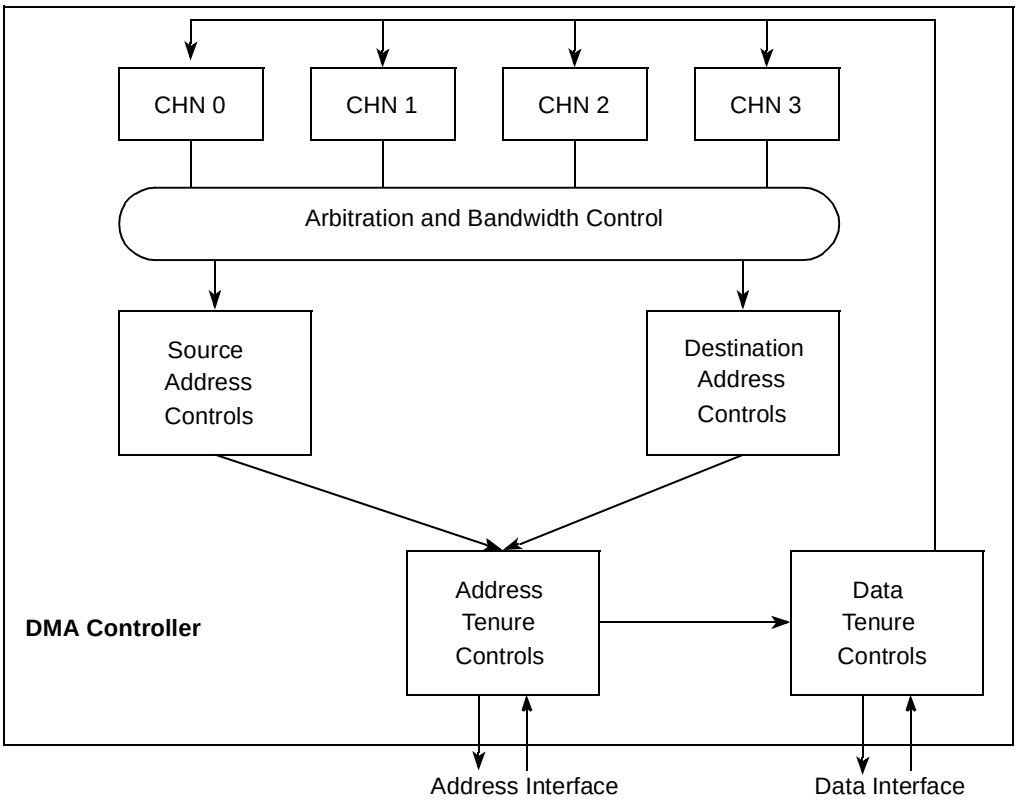


Figure 15-1. DMA Block Diagram

15.1.2 Overview

The DMA controller has four high-speed DMA channels. Both the e500 core and external devices can initiate DMA transfers. All channels are capable of complex data movement and advanced

transaction chaining. Figure 15-1 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.

15.1.3 Features

The DMA controller offers the following features:

- Four high-speed/high-bandwidth channels accessible by local and remote masters
- Basic DMA operation modes (direct, simple chaining)
- Extended DMA operation modes (advanced chaining and stride capability)
- Cascading descriptor chains
- Misaligned transfers
- Programmable bandwidth control between channels
- Up to 256 bytes for DMA sub-block transfers to maximize performance over the RapidIO interface
- Three priority levels supported for source and destination transactions
- Interrupt on error and completed segment, list, or link
- Externally-controlled transfer using $\overline{\text{DMA\_DREQ}}$, $\overline{\text{DMA\_DACK}}$, and $\overline{\text{DMA\_DDONE}}$

15.1.4 Modes of Operation

The MPC8540 has two modes of operation: basic and extended. Basic mode is the DMA legacy mode. It does not support advanced features. Extended mode supports advanced features like striding and flexible descriptor structures.

These two basic modes allow users to initiate and end DMA transfers in various ways. Table 15-1 summarizes the relationship between the modes and the following features:

- Direct mode. No descriptors are involved. Software must initialize the required fields as described in Table 15-2 before starting a transfer.
- Chaining mode. Software must initialize descriptors in memory and the required fields as described in Table 15-2 before starting a transfer.
- Single-write start mode. The DMA process can be started by using a single-write command to either the descriptor address register in one of the chaining modes or the source/destination address registers in one of the direct modes.

- External control capability: This allows an external agent to start, pause, and check the status of a DMA transfer which has already been initialized.
- Channel continue capability: The channel continue capability allows software the flexibility of having the DMA controller start with descriptors that have already been programmed while software continues to build more descriptors in memory.
- Channel abort capability: The software can abort a previously initiated transfer by setting the bit MR*n*[CA]. The DMA controller terminates all outstanding transfers initiated by the channel without generating any errors before entering an idle state.

Table 15-1. Relationship of Modes and Features

Mode	Mode with 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 15-2 describes bit settings required for each DMA mode of operation

Table 15-2. 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 15-2. 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 15.4, “Functional Description,” for details on these modes.

Figure 15-2 shows the general DMA operational flow chart.

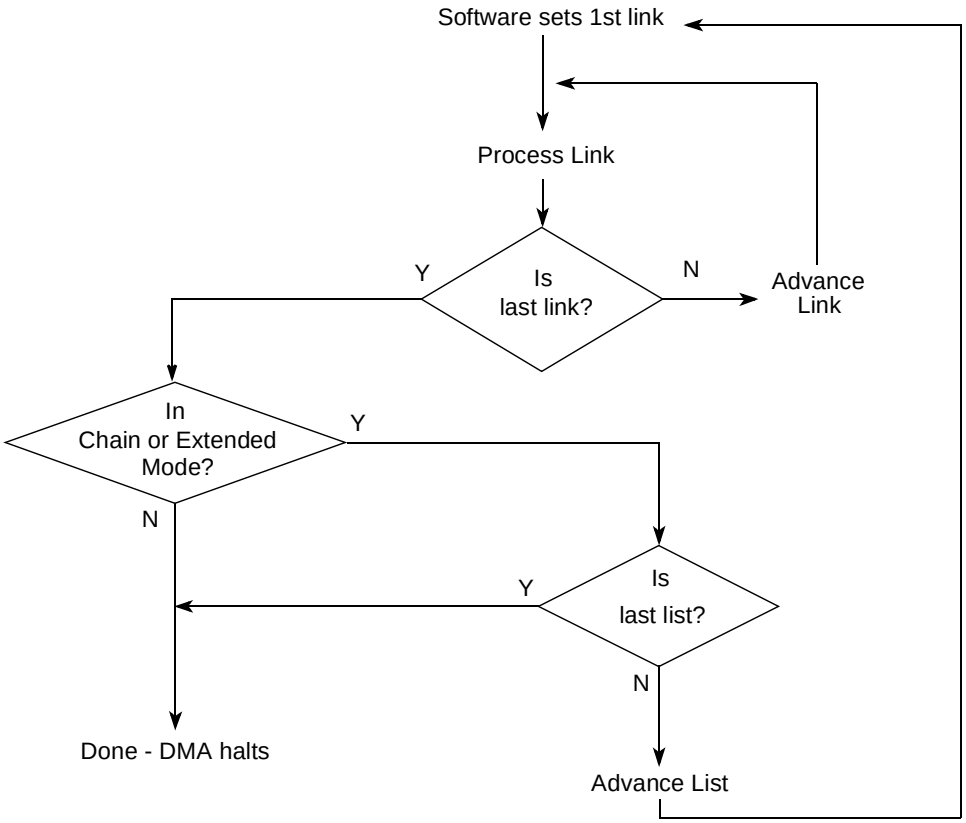


Figure 15-2. DMA Operational Flow Chart

15.2 External Signal Descriptions

This section describes the DMA signals.

15.2.1 Signal Overview

Figure 15-3 summarizes the DMA controller signals.

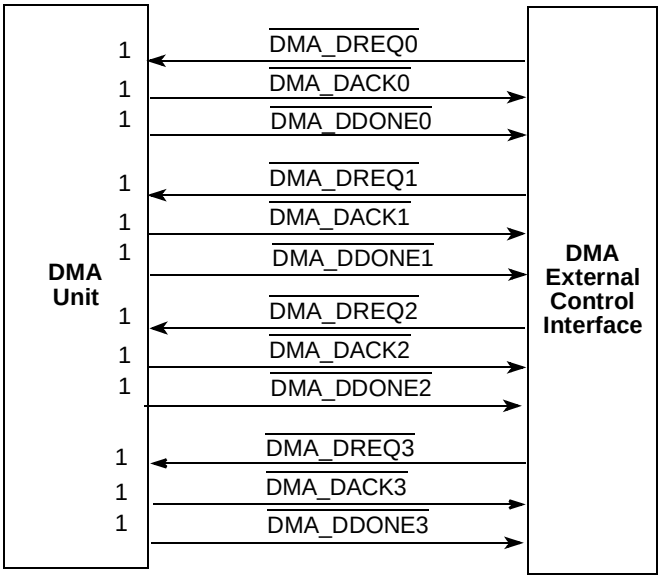


Figure 15-3. DMA Signal Summary

Note that the three DMA signals for DMA channel 3 are multiplexed with the IRQ9–11 signals on the MPC8540 device. These functions are mutually exclusive and the active function is specified in the PMUXCR register of the global utilities block as described in Section 18.4.1.10, "Alternate Function Signal Multiplex Control Register (PMUXCR)."

15.2.2 Detailed Signal Descriptions

Table 15-3 describes the DMA signals.

Table 15-3. DMA Signals—Detailed Signal Descriptions

Signal	I/O	Description	
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 DMA_DREQ*n* causes MR*n*[CS] to be set, thereby activating the corresponding DMA channel.	
		State Meaning	Asserted—Assertion of DMA_DREQ*n* while 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 DMA_DACK*n* is asserted results in an illegal condition Negated—Negation while DMA_DACK*n* is asserted has no effect. Negation before the assertion of 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 DMA_DACK*n*
DMA_DACK*n*	O	DMA acknowledge. Indicates that a DMA transfer is currently in progress	
		State Meaning	Asserted—Indicates that a DMA transfer is currently in progress. Asserted after the assertion of DMA_DREQ to indicate the start of a transfer Negated—Negated after finishing a complete transfer or after entering a paused state if MR*n*[EMP_EN] is set
		Timing	Assertion—Asynchronous assertion; asserted for more than three system clocks Negation—Asynchronous negation; negated for more than three system clocks
DMA_DDONE*n*	O	DMA done. Indicates that a DMA transfer is complete.	
		State Meaning	Asserted—Indicates transfer completion. SRn[CB] is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface. Negated—Indicates that the current transfer is in process
		Timing	Assertion—Always asserts asynchronously after the negation of the final DMA_DACK*n* to indicate completion of a transfer. For a paused transfer, DMA_DDONE*n* is asserted asynchronously after the negation of the final DMA_DACK*n*. Negation—Negated asynchronously after the assertion of DMA_DREQ*n* for the next transfer

15.3 Memory Map/Register Definition

This section provides a detailed description of all accessible DMA memory and registers. The descriptions include individual bit level descriptions and reset states of each register.

15.3.1 Module Memory Map

Figure 15-4 shows the address map of DMA register space.

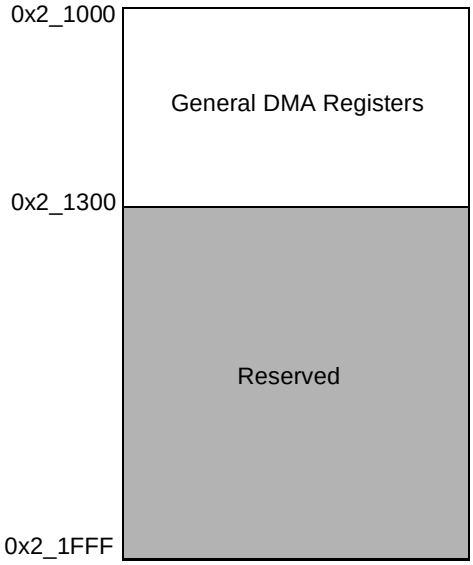


Figure 15-4. DMA Register Space (Memory-Mapped)

Table 15-4 lists the DMA registers.

Table 15-4. DMA Register Summary

Offset	Register	Access	Reset	Section/Page
General Registers				
0x2_1100	MR0—DMA mode register 0	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1104	SR0—DMA status register 0	Special	0x0000_0000	15.3.2.2/15-13
0x2_1108	Reserved	—	—	—
0x2_110C	CLNDAR0—DMA current link descriptor address register 0	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1110	SATR0—DMA source attributes register 0	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1114	SAR0—DMA source address register 0	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1118	DATR0—DMA destination attributes register 0	R/W	0x0000_0000	15.3.2.6/15-19
0x2_111C	DAR0—DMA destination address register 0	R/W	0x0000_0000	15.3.2.7/15-21
0x2_1120	BCR0—DMA byte count register 0	R/W	0x0000_0000	15.3.2.8/15-23
0x2_1124	Reserved	—	—	—
0x2_1128	NLNDAR0—DMA next link descriptor address register 0	R/W	0x0000_0000	15.3.2.9/15-23
0x2_112C–0x2_1130	Reserved	—	—	—

Table 15-4. 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	15.3.2.10/15-24
0x2_1138	Reserved	—	—	—
0x2_113C	NLSDAR0—DMA next list descriptor address register 0	R/W	0x0000_0000	15.3.2.11/15-25
0x2_1140	SSR0—DMA source stride register 0	R/W	0x0000_0000	15.3.2.12/15-25
0x2_1144	DSR0—DMA destination stride register 0	R/W	0x0000_0000	15.3.2.13/15-26
0x2_1148–0x2_117C	Reserved	—	—	—
0x2_1180	MR1—DMA mode register 1	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1184	SR1—DMA status register 1	Special	0x0000_0000	15.3.2.2/15-13
0x2_1188	Reserved	—	—	—
0x2_118C	CLNDAR1—DMA current link descriptor address register 1	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1190	SATR1—DMA source attributes register 1	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1194	SAR1—DMA source address register 1	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1198	DATR1—DMA destination attributes register 1	R/W	0x0000_0000	15.3.2.6/15-19
0x2_119C	DAR1—DMA destination address register 1	R/W	0x0000_0000	15.3.2.7/15-21
0x2_11A0	BCR1—DMA byte count register 1	R/W	0x0000_0000	15.3.2.8/15-23
0x2_11A4	Reserved	—	—	—
0x2_11A8	NLNDAR1—DMA next link descriptor address register 1	R/W	0x0000_0000	15.3.2.9/15-23
0x2_11AC–0x2_11B0	Reserved	—	—	—
0x2_11B4	CLSDAR1—DMA current list alternate base descriptor address register 1	R/W	0x0000_0000	15.3.2.10/15-24
0x2_11B8	Reserved	—	—	—
0x2_11BC	NLSDAR1—DMA next list descriptor address register 1	R/W	0x0000_0000	15.3.2.11/15-25
0x2_11C0	SSR1—DMA source stride register 1	R/W	0x0000_0000	15.3.2.12/15-25
0x2_11C4	DSR1—DMA destination stride register 1	R/W	0x0000_0000	15.3.2.13/15-26
0x2_11C8–0x2_11FC	Reserved	—	—	—
0x2_1200	MR2—DMA mode register 2	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1204	SR2—DMA status register 2	Special	0x0000_0000	15.3.2.2/15-13
0x2_1208	Reserved	—	—	—
0x2_120C	CLNDAR2—DMA current link descriptor address register 2	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1210	SATR2—DMA source attributes register 2	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1214	SAR2—DMA source address register 2	R/W	0x0000_0000	15.3.2.5/15-18

Table 15-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x2_1218	DATR2—DMA destination attributes register 2	R/W	0x0000_0000	15.3.2.6/15-19
0x2_121C	DAR2—DMA destination address register 2	R/W	0x0000_0000	15.3.2.7/15-21
0x2_1220	BCR2—DMA byte count register 2	R/W	0x0000_0000	15.3.2.8/15-23
0x2_1224	Reserved	—	—	—
0x2_1228	NLNDAR2—DMA next link descriptor address register 2	R/W	0x0000_0000	15.3.2.9/15-23
0x2_122C–0x2_1230	Reserved	—	—	—
0x2_1234	CLSDAR2—DMA current list alternate base descriptor address register 2	R/W	0x0000_0000	15.3.2.10/15-24
0x2_1238	Reserved	—	—	—
0x2_123C	NLSDAR2—DMA next list descriptor address register 2	R/W	0x0000_0000	15.3.2.11/15-25
0x2_1240	SSR2—DMA source stride register 2	R/W	0x0000_0000	15.3.2.12/15-25
0x2_1244	DSR2—DMA destination stride register 2	R/W	0x0000_0000	15.3.2.13/15-26
0x2_1248–0x2_127C	Reserved	—	—	—
0x2_1280	MR3—DMA mode register 3	R/W	0x0000_0000	15.3.2.1/15-10
0x2_1284	SR3—DMA status register 3	Special	0x0000_0000	15.3.2.2/15-13
0x2_1288	Reserved	—	—	—
0x2_128C	CLNDAR3—DMA current link descriptor address register 3	R/W	0x0000_0000	15.3.2.3/15-14
0x2_1290	SATR3—DMA source attributes register 3	R/W	0x0000_0000	15.3.2.4/15-16
0x2_1294	SAR3—DMA source address register 3	R/W	0x0000_0000	15.3.2.5/15-18
0x2_1298	DATR3—DMA destination attributes register 3	R/W	0x0000_0000	15.3.2.6/15-19
0x2_129C	DAR3—DMA destination address register 3	R/W	0x0000_0000	15.3.2.7/15-21
0x2_12A0	BCR3—DMA byte count register 3	R/W	0x0000_0000	15.3.2.8/15-23
0x2_12A4	Reserved	—	—	—
0x2_12A8	NLNDAR3—DMA next link descriptor address register 3	R/W	0x0000_0000	15.3.2.9/15-23
0x2_12AC–0x2_12B0	Reserved	—	—	—
0x2_12B4	CLSDAR3—DMA current list alternate base descriptor address register 3	R/W	0x0000_0000	15.3.2.10/15-24
0x2_12B8	Reserved	—	—	—
0x2_12BC	NLSDAR3—DMA next list descriptor address register 3	R/W	0x0000_0000	15.3.2.11/15-25
0x2_12C0	SSR3—DMA source stride register 3	R/W	0x0000_0000	15.3.2.12/15-25
0x2_12C4	DSR3—DMA destination stride register 3	R/W	0x0000_0000	15.3.2.13/15-26

Table 15-4. DMA Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0x2_12C8–0x2_12FC	Reserved	—	—	—
0x2_1300	DGSR—DMA general status register	Read	0x0000_0000	15.3.2.14/15-26

15.3.2 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.

15.3.2.1 Mode Registers (MR*n*)

The mode register allows software to start a DMA transfer and to control various DMA transfer characteristics. Figure 15-5 describes the MR*n*.

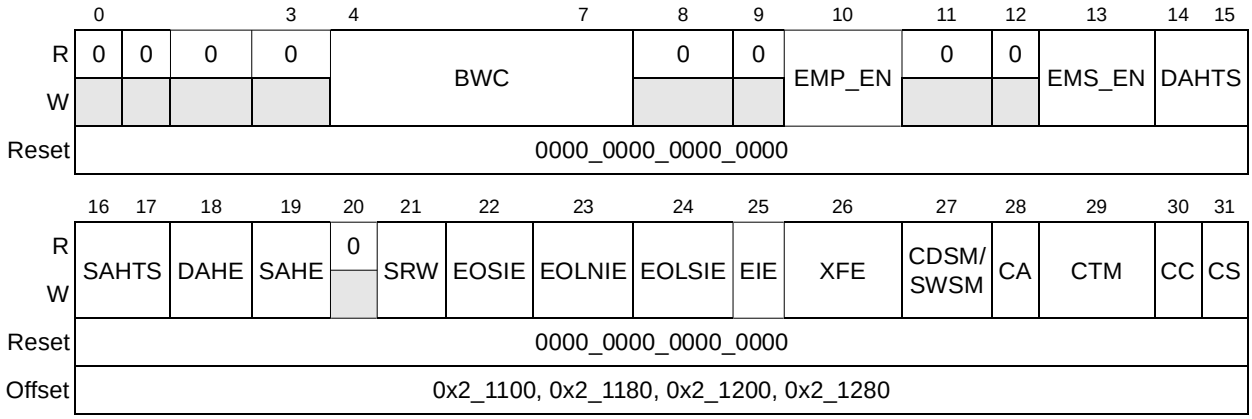


Figure 15-5. DMA Mode Registers (MR*n*)

Table 15-5 describes the MR*n* fields.

Table 15-5. 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 15-5. MR*n* Field Descriptions (continued)

Bits	Name	Description
20	—	Reserved
21	SRW	Single register write (Direct mode only; reserved for chaining mode.) 0 Normal operation. 1 Enable a write to the source address register to simultaneously set MR*n*[CS], starting a DMA transfer, when MR*n*[CDSM/SWSM] is also set. Setting this bit and clearing 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 15-5. MR*n* Field Descriptions (continued)

Bits	Name	Description
30	CC	Channel continue. This bit applies only to chaining mode and is cleared by hardware after the first descriptor read when continuing a transfer. This bit is reserved for external master mode. 0 No effect. 1 The DMA transfer restarts the transferring process starting at the current descriptor address.
31	CS	Channel start. This bit is also automatically set by hardware during single-write start mode and external master start enable mode. 0 Halt the DMA process if channel is busy (SR*n*[CB] is set). No effect if the channel is not busy. 1 Start the DMA process if channel is not busy (CB is cleared). If the channel was halted (CS = 0 and CB = 1), the transfer continues from the point at which it was halted.

15.3.2.2 Status Registers (SR*n*)

The status registers, shown in Figure 15-6, 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 15-6. Status Registers (SR*n*)

Table 15-6 describes the bits of the SR*n*.

Table 15-6. SR*n* Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TE	Transfer error (Bit reset, write 1 to clear) 0 No error condition during the DMA transfer. 1 Error condition during the DMA transfer. See Section 15.4.3, “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 15-6. 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)

15.3.2.3 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 15-7, software must initialize this register to point to the first link descriptor in memory.

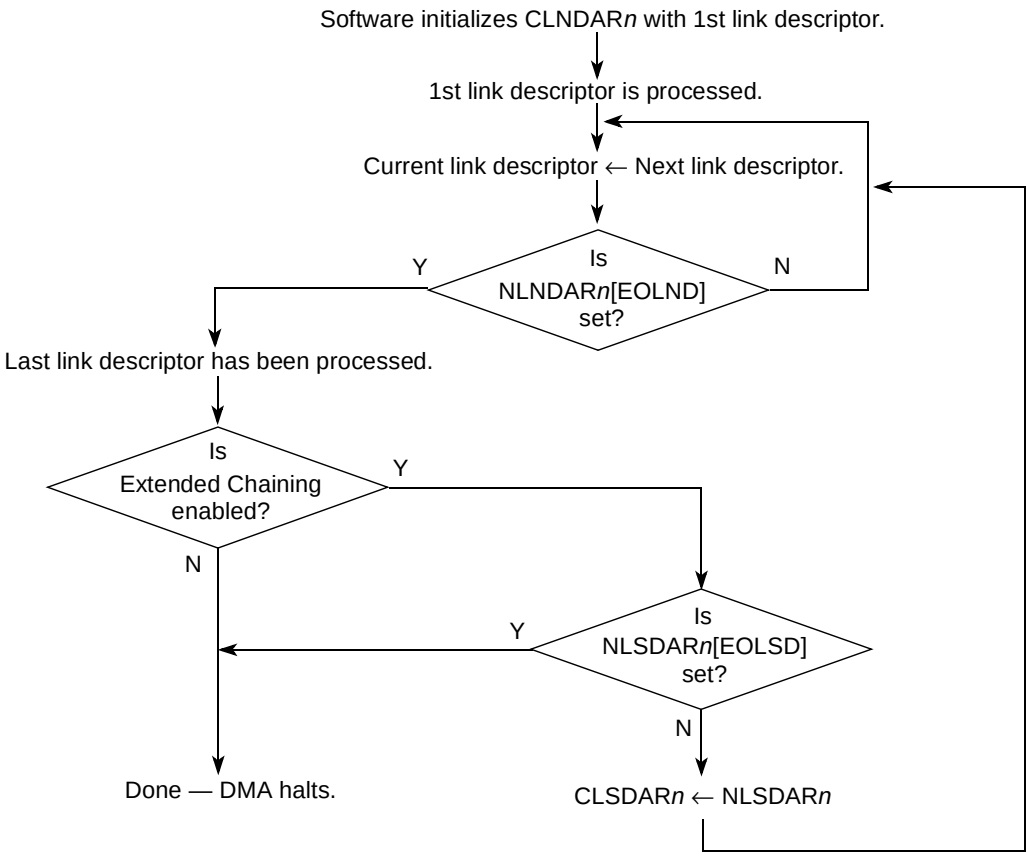


Figure 15-7. Basic Chaining Mode Flow Chart

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 15-8 shows CLNDAR*n*.

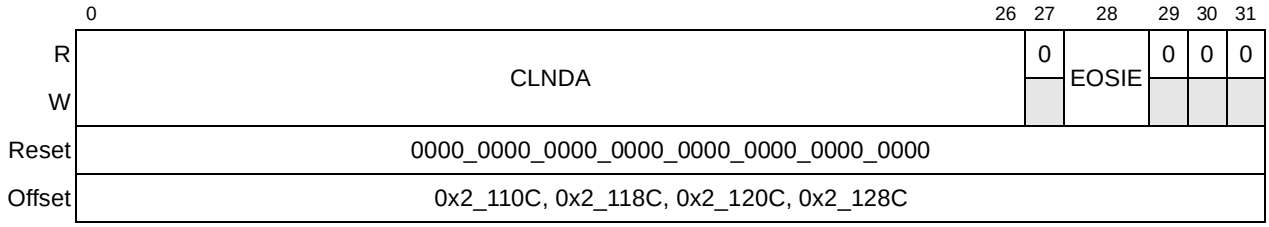


Figure 15-8. Current Link Descriptor Address Registers (CLNDAR*n*)

Table 15-7 describes the fields of the CLNDAR*n*.

Table 15-7. CLNDAR*n* Field Descriptions

Bits	Name	Description
0–26	CLNDA	Current link descriptor address. Contains the current descriptor address of the buffer descriptor in memory. The descriptor must be aligned to a 32-byte boundary.
27	—	Reserved
28	EOSIE	End-of-segment interrupt enable 0 Do not generate an interrupt upon completion of the current DMA transfer for the current link descriptor. 1 Generate an interrupt upon completion of the current DMA transfer for the current link descriptor.
29–31	—	Reserved

15.3.2.4 Source Attributes Registers (SATR*n*)

The source attributes registers, shown in Figure 15-9, 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.

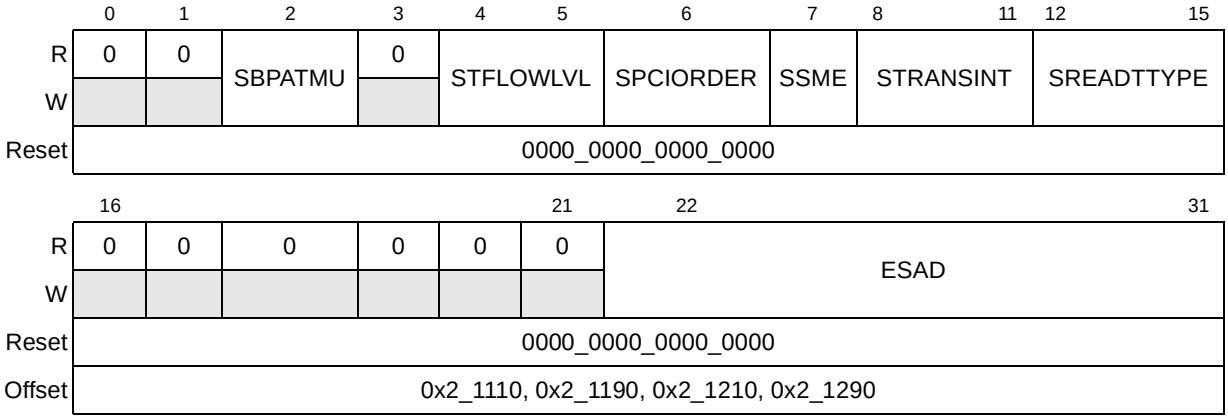


Figure 15-9. Source Attributes Registers (SATR*n*)

Table 15-8 describes the fields of the SATR*n*.

Table 15-8. 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 15-8. 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.

15.3.2.5 Source Address Registers (SAR*n*)

The source address registers, shown in Figure 15-10, 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 15-10. Source Address Registers (SAR*n*)

Table 15-9 describes the field of the SAR*n*.

Table 15-9. 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.

15.3.2.5.1 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 15-11 describes the SAR*n*.

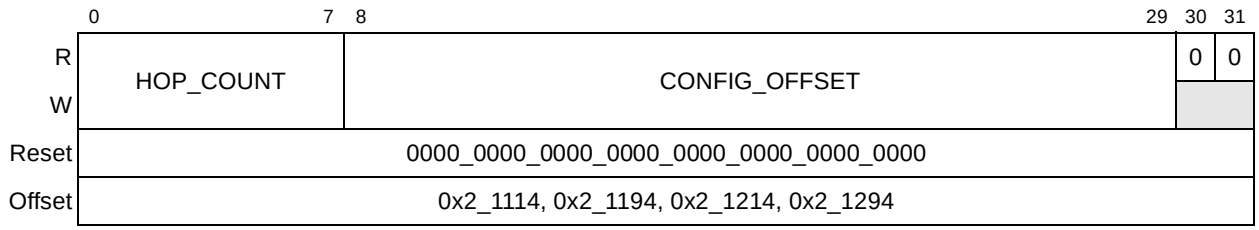


Figure 15-11. Source Address Registers for RapidIO Maintenance Reads (SAR*n*)

Table 15-10 describes the fields of the SAR*n*.

Table 15-10. 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

15.3.2.6 Destination Attributes Registers (DATR*n*)

The destination attributes registers, shown in Figure 15-12, 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.

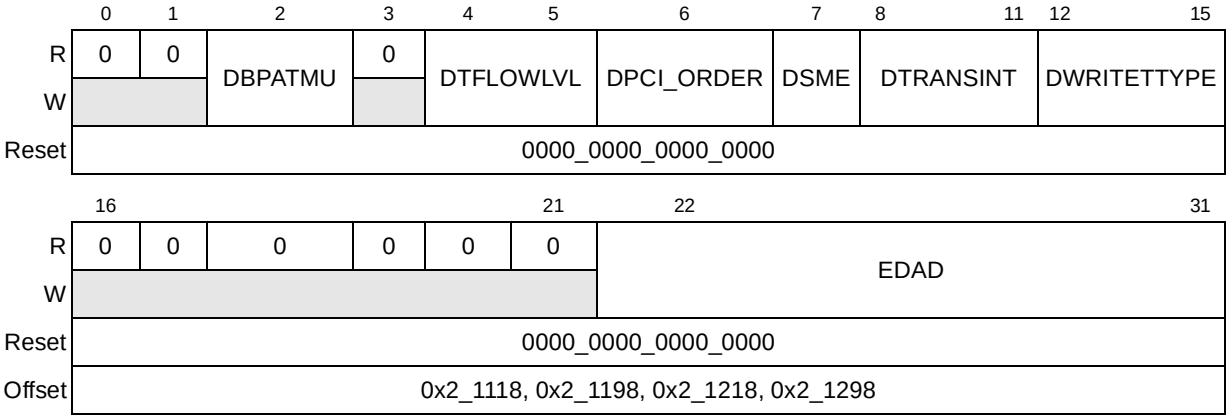


Figure 15-12. Destination Attributes Registers (DATR*n*)

Table 15-11 describes the fields of the DATR*n*.

Table 15-11. DATR*n* Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	DBPATMU	Bypass ATMU for this DMA operation 0 Route the operation through the ATMU outbound windows. DATR*n*[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 15-11. 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 1000–1111Reserved • Transaction type to run on local address space—Used even in non-ATMU bypass mode 0000–0011Reserved 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.

15.3.2.7 Destination Address Registers (DAR*n*)

The destination address registers, shown in Figure 15-13, 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.

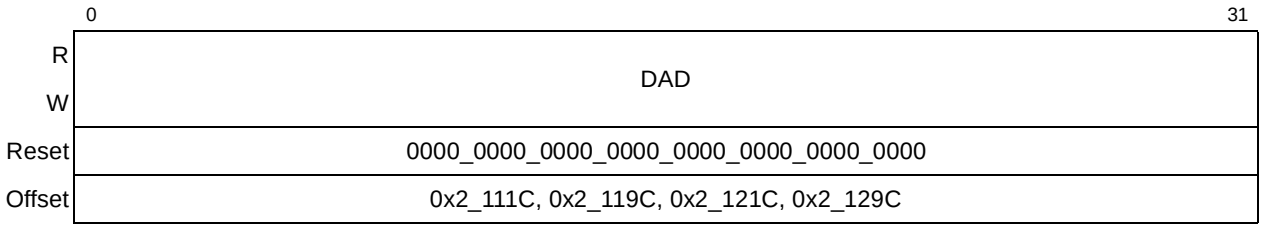


Figure 15-13. Destination Address Registers (DAR*n*)

Table 15-12 describes the field of the DAR*n*.

Table 15-12. 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.

15.3.2.7.1 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 15-14 shows the DAR*n*.

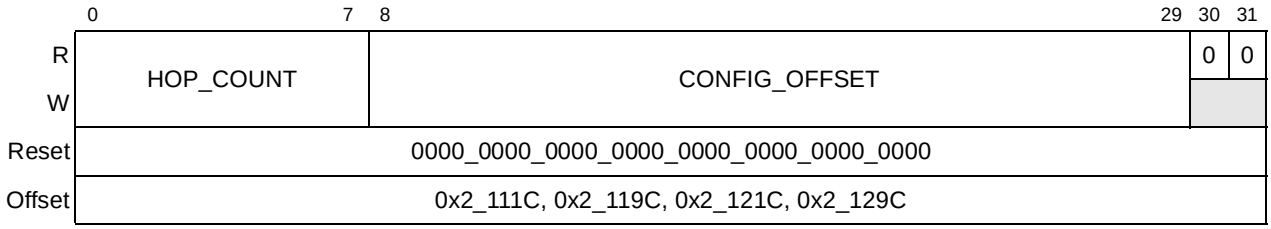


Figure 15-14. Destination Address Registers for RapidIO Maintenance Writes (DAR*n*)

Table 15-13 describes the fields of the DAR*n*.

Table 15-13. 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

15.3.2.8 Byte Count Registers (BCR*n*)

The byte count register, shown in Figure 15-15, contains the number of bytes to transfer.

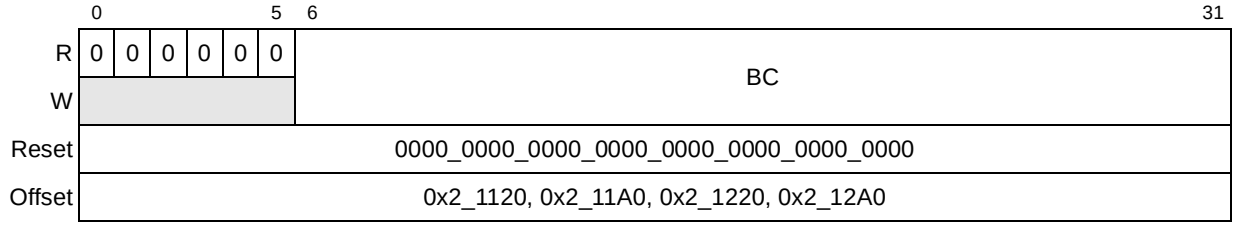


Figure 15-15. Byte Count Registers (BCR*n*)

Table 15-14 describes the fields of the BCR*n*.

Table 15-14. BCR*n* Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6–31	BC	Byte count. Contains the number of bytes to transfer. The value in this register is decremented after each DMA read operation. The maximum transfer size is 2^{26} – 1 Bytes.

15.3.2.9 Next Link Descriptor Address Registers (NLNDAR*n*)

The next link descriptor address registers, shown in Figure 15-16, 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.

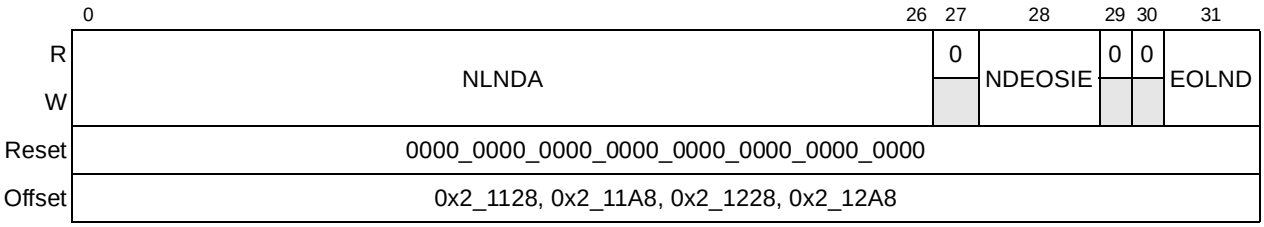


Figure 15-16. Next Link Descriptor Address Registers (NLNDAR*n*)

Table 15-15 describes the fields of the NLNDAR*n* registers.

Table 15-15. 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 15-15. 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.

15.3.2.10 Current List Descriptor Address Registers (CLSDAR*n*)

The current list descriptor address registers, shown in Figure 15-17, 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.

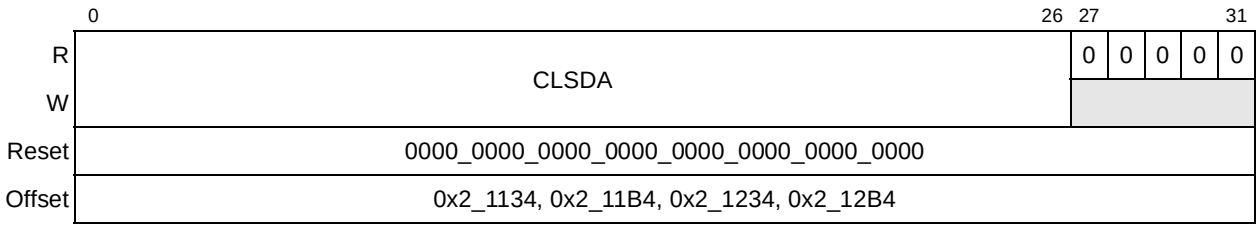


Figure 15-17. Current List Descriptor Address Registers (CLSDAR*n*)

Table 15-16 describes the fields of the CLSDAR*n*.

Table 15-16. 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

15.3.2.11 Next List Descriptor Address Registers (NLSDAR*n*)

The next list descriptor address register, shown in Figure 15-18, 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.

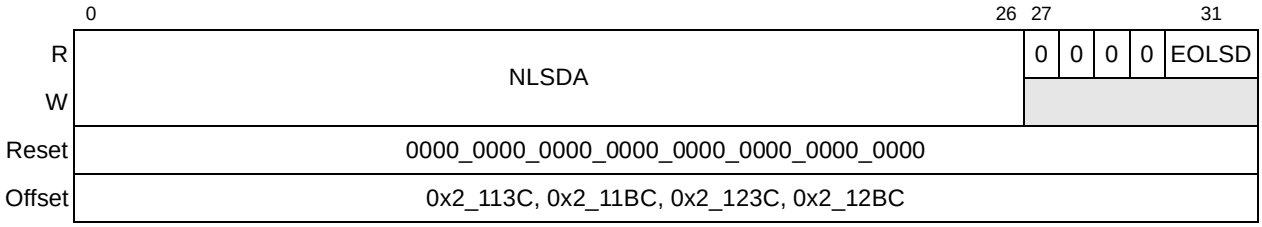


Figure 15-18. Next List Descriptor Address Registers (NLSDAR*n*)

Table 15-17 describes the fields of the NLSDAR*n*.

Table 15-17. 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.

15.3.2.12 Source Stride Registers (SSR*n*)

The source stride register, shown in Figure 15-19, 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.

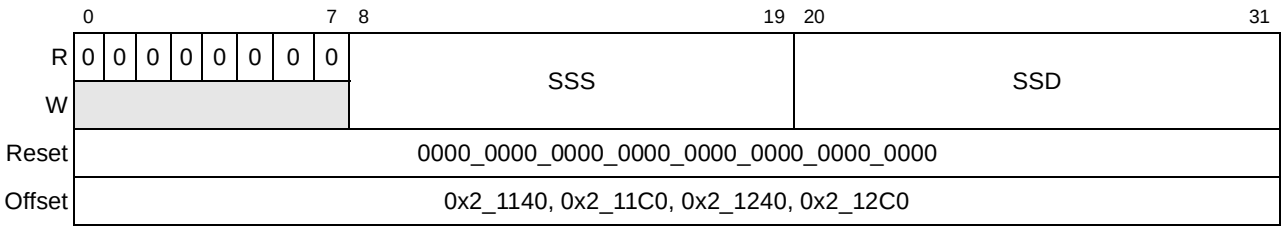


Figure 15-19. Source Stride Registers (SSR*n*)

Table 15-18 describes the fields of the SSR*n*.

Table 15-18. 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.

15.3.2.13 Destination Stride Registers (DSR*n*)

The destination stride register contains the stride size, and distance. Note that the destination stride information is loaded when a new list descriptor is read from memory. Therefore, the destination stride register is applicable for all link descriptors in the new list. Changing the destination stride information for a link requires that a new list be generated. Figure 15-20 describes the DSR*n*.

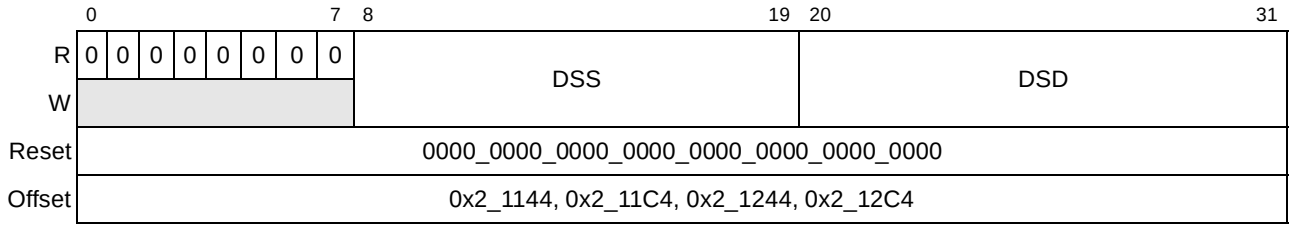


Figure 15-20. Destination Stride Registers (DSR*n*)

Table 15-19 describes the fields of the DSR*n*.

Table 15-19. 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.

15.3.2.14 DMA General Status Register (DGSR)

The DMA general status register combines all of the status bits from each channel into one register, including the enhanced DMA channel. This register is read-only. Figure 15-21 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 15-21. DMA General Status Register (DGSR)

Table 15-20 describes the fields of the DGSR.

Table 15-20. 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 15-20. 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

15.4 Functional Description

This section describes the function of the the DMA controller.

15.4.1 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 15-21 on page 15-36 for more complete descriptions of the channel states and state transitions.

15.4.1.1 Basic DMA Mode Transfer

This mode is primarily included for backward compatibility with existing DMA controllers which use a simple programming model. This is the default mode out of reset. The different modes of operation under the basic mode are explained in the following sections.

15.4.1.1.1 Basic Direct Mode

In basic direct mode, the DMA controller does not read descriptors from memory, but instead uses the current parameters programmed in the DMA registers to start the DMA transfer. Software is responsible for initializing 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 15-21), 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.

15.4.1.1.2 Basic Direct Single-Write Start Mode

In basic direct single-write start mode, the DMA controller does not read descriptors from memory, but instead uses the current parameters programmed in the DMA registers to start the DMA transfer. Software is responsible for initializing the 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 15-21), 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.

15.4.1.1.3 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 15.4.4, "DMA Descriptors," for more information on descriptors.
2. Poll the channel state (see Table 15-21), 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.

15.4.1.1.4 Basic Chaining Single-Write Start Mode

Basic chaining single-write start mode allows a chain to be started by writing the current link descriptor address register (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 15.4.4, "DMA Descriptors," for more information on descriptors.
3. Poll the channel state (see Table 15-21), 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.

15.4.1.2 Extended DMA Mode Transfer

The extended DMA mode also operates in chaining and direct mode. It offers additional capability over the basic mode by supporting striding and a more flexible descriptor structure. This additional functionality also requires a new and more complex programming model. The extended DMA mode is activated by setting MR*n*[XFE].

15.4.1.2.1 Extended Direct Mode

Extended direct mode has the same functionality as basic direct mode with the addition of stride capabilities. The bit settings are the same as in direct mode with the exception of the 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*.

15.4.1.2.2 Extended Direct Single-Write Start Mode

Extended direct single-write start mode has the same functionality as the basic direct single-write start mode with the addition of stride capabilities. The bit settings are also the same with the exception of 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*.

15.4.1.2.3 Extended Chaining Mode

In extended chaining mode, the software must first build list and link descriptor segments in memory. Then 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 15.4.4, “DMA Descriptors,” for more information on descriptors.

2. Poll the channel state (see Table 15-21), 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.

15.4.1.2.4 Extended Chaining Single-Write Start Mode

In the extended mode, the single-write start feature allows a chain to be started by writing the current list descriptor pointer. Setting MR*n*[CDSM/SWSM] causes MR*n*[CS] to be set automatically when 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 15-21), 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.

15.4.1.3 External Control Mode Transfer

An external control can be used to control all DMA channels by setting MR*n*[EMS_EN]. The external control can control the DMA channel in the following transfer modes:

- Basic direct
- Basic chaining
- Extended direct
- Extended chaining

The external control and the DMA controller use a well defined protocol to communicate. The external control can start or pause a DMA transfer. The DMA controller acknowledges a DMA transfer in progress and also indicates a transfer completion.

The pause feature can be enabled by setting 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 15-4. The timing diagram of the external control interface is shown in Figure 15-22.

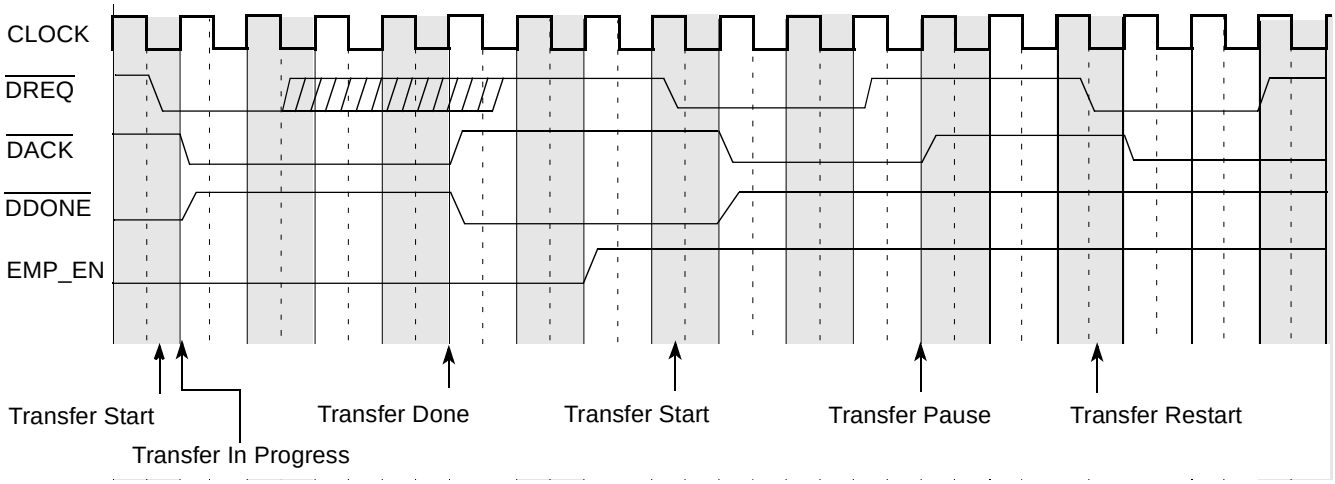


Figure 15-22. External Control Interface Timing

15.4.1.4 Channel Continue Mode for Cascading Transfer Chains

The channel continue mode (enabled when 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.

15.4.1.4.1 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.

15.4.1.4.2 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.

15.4.1.5 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.

15.4.1.6 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.

15.4.1.7 Channel State

Table 15-21 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 15-21. 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.

15.4.1.8 Illustration of Stride Size and Stride Distance

If operating in stride mode, the stride size defines the amount of data to transfer before jumping to the next quantity of data as specified by the stride distance. The stride distance is added to the current base address to point to the next quantity of data to be transferred. Figure 15-23 illustrates the stride size and distance parameters. As shown, each time the stride distance is added to the base address, the resulting address becomes the new base address. This sequence repeats until the amount of data transferred equals the transfer size.

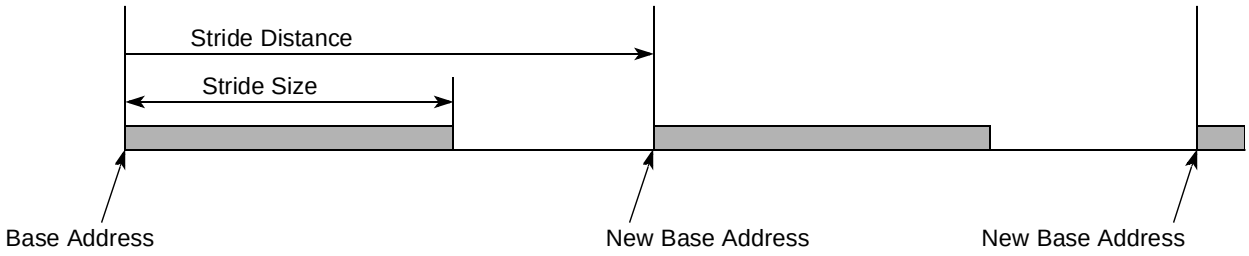


Figure 15-23. Stride Size and Stride Distance

15.4.2 DMA Transfer Interfaces

The DMA can be used to achieve data transfers across the entire memory map.

15.4.3 DMA Errors

On a transfer error (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.

15.4.4 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 15-24. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor. Table 15-22 summarizes the DMA list descriptors.

Table 15-22. 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 15-23 summarizes the DMA link descriptors.

Table 15-23. 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 15-24 describes the DMA transaction flow.

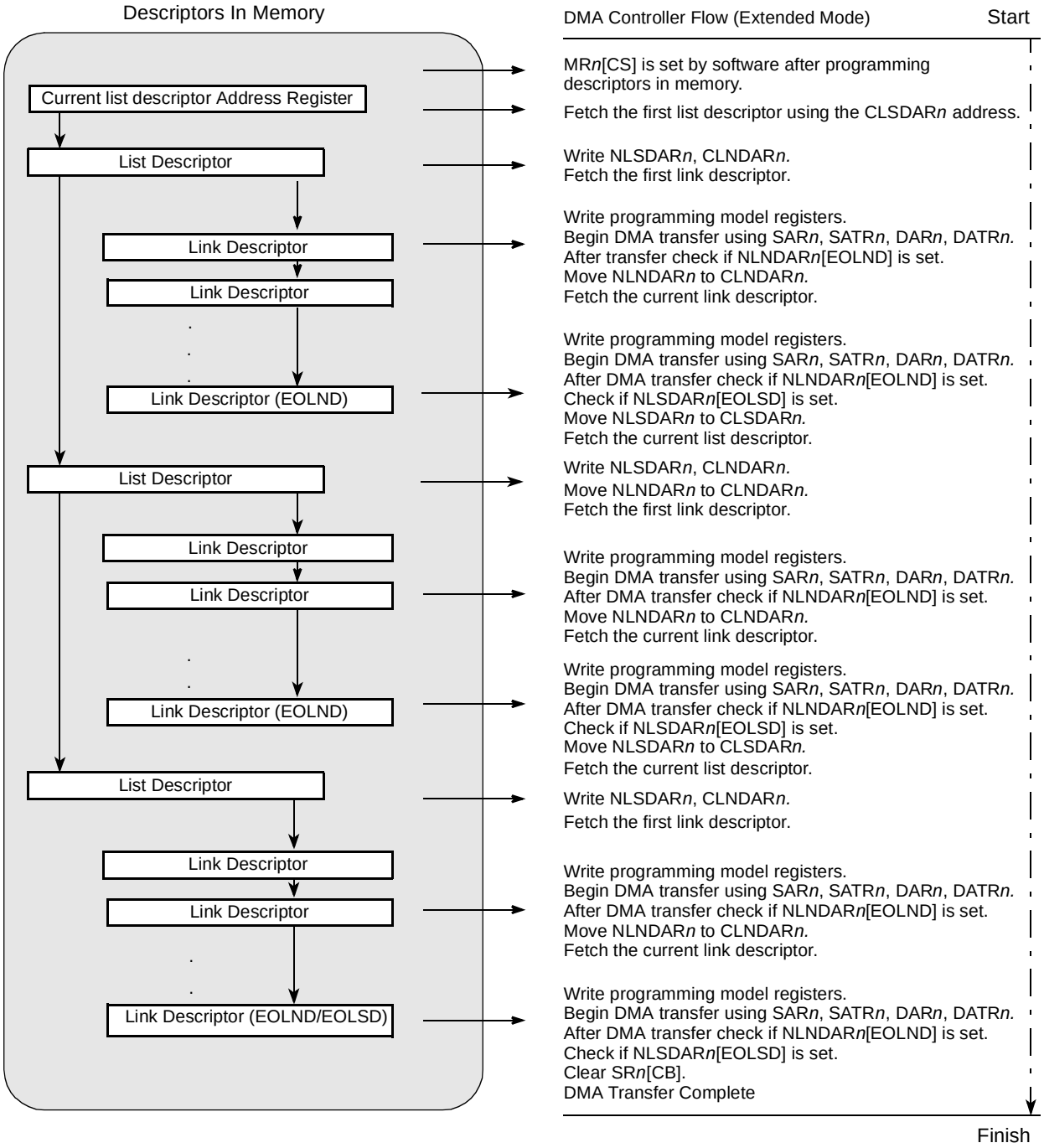


Figure 15-24. DMA Transaction Flow with DMA Descriptors

Figure 15-25 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 15-25. List Descriptor Format

Figure 15-26 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 15-26. Link Descriptor Format

15.4.5 Limitations and Restrictions

This section addresses some of the limitations and restrictions of the DMA controller and is intended to help software maximize the DMA performance and avoid DMA programming errors.

The limitations of the DMA controller are the following:

- Due to the limited number of buffers that the DMA controller can use, stride sizes less than 64 bytes should be avoided. Maximum utilization is obtained from strides greater than or equal to 256-bytes. However, small stride sizes can be used for scatter-gather functions.
- Coherent reads or writes are broken up into cache line accesses in the DMA.

The DMA controller restrictions are 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.

15.5 DMA System Considerations

Figure 15-27 shows the most important data paths within the MPC8540. Many of these paths are served by captive DMA controllers and virtually all can be served by the general purpose four-channel DMA controller. This section provides information about how to make most effective use of these DMA channels including the following topics:

- DMA transaction initiators (masters)
- DMA targets, that is, data sources and destinations

- Transparency of the bus interfaces to DMA operations
- What is useful as opposed to what is possible. For example, it is possible to address any internal register via the internal control bus, which means configuration and control registers can be DMA source or destination targets. However, the typical use of DMA functionality is to reduce host processor loading by moving large amounts of data with minimal CPU involvement. Using a general-purpose DMA controller to load small amounts of configuration data only makes sense in special circumstances (perhaps during system boot, for example).

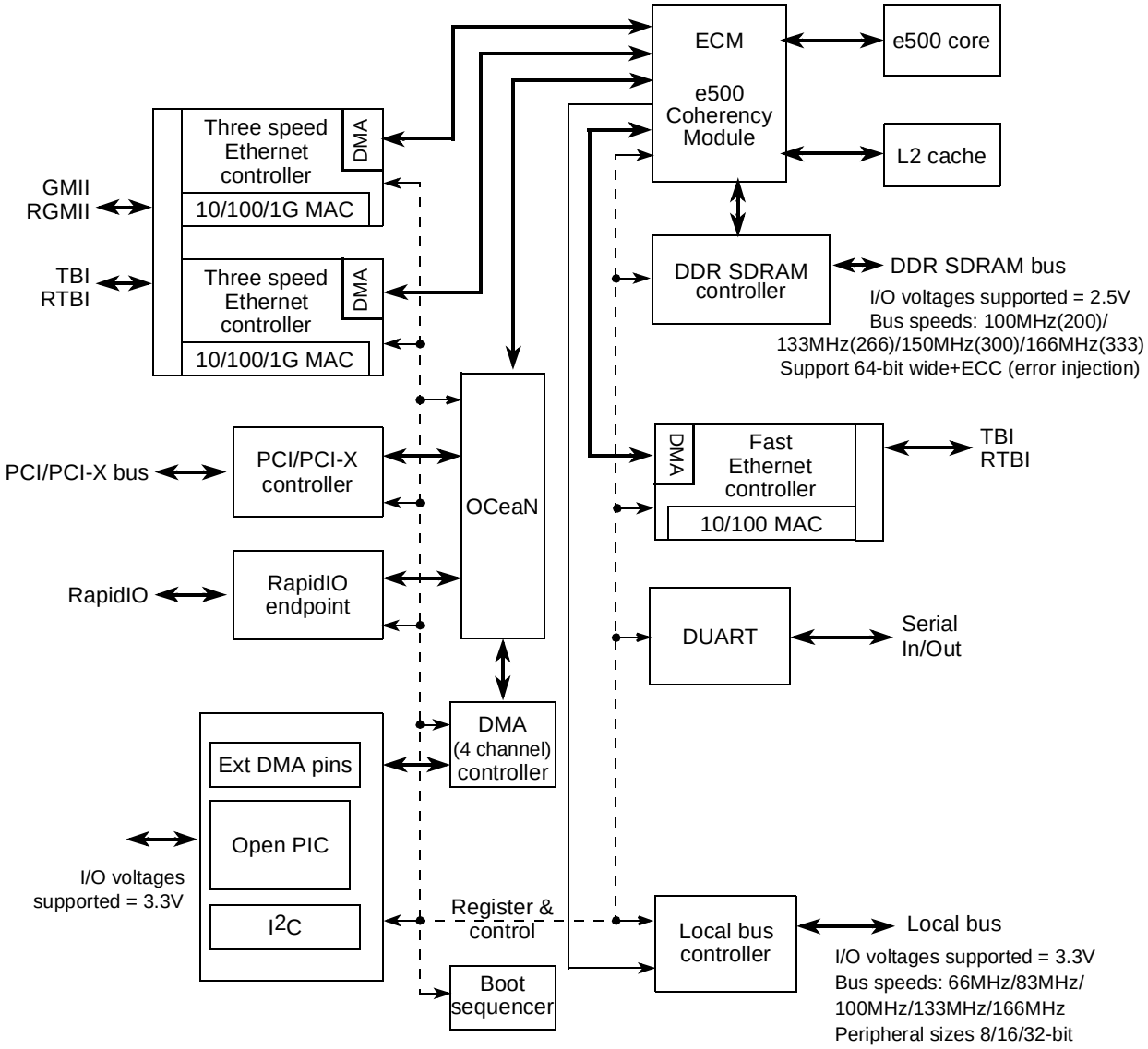


Figure 15-27. DMA Data Paths

Table 15-24 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 15-24. DMA Paths

DMA Controllers		On-Chip Targets		Off-Chip Targets					
		L2	Configuration Registers[1]	DDR	Local Bus	PCI/ PCI-X	RapidIO	DUART FIFO	Ethernet Buffers
On-chip	Ethernet[2]	Y	NS	Y	Y	Y	Y	Y	Y
	4 Channel[3]	Y	NR	Y	Y	Y	Y	Y	NS
Off-chip	PCI-X Controller	Y	NS	Y	Y	—	Y	Y	Y
	RapidIO Controller	Y	NS	Y	Y	Y	—	Y	Y

1 Includes I^2C data register.

2 Captive resource. Not available to external masters.

3 Can serve external masters.

15.5.1 Unusual DMA Scenarios

The following is a description of unusual DMA paths including explanations of why some functional blocks cannot serve as DMA targets. The following topics are addressed:

- Transaction initiators (masters)
- DMA targets, that is, data sources or destinations
- Transparency of the bus controllers to DMA transactions
- What is useful as opposed to what is possible. For example, any register can be addressed through an internal control bus, which means configuration and control registers can be DMA targets.

15.5.1.1 DMA to e500 Core

The L1 cache 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.

15.5.1.2 DMA to Ethernet

The Ethernet controllers cannot serve as DMA targets because they have no suitable internal memory for this purpose. The Ethernet controllers have dedicated DMA channels to move data between the external transmit and receive buffers and the internal packet buffer. This dedicated channel is the only DMA service to the internal packet buffers.

However, Ethernet ports can serve as DMA targets by using a general-purpose DMA controller to access the transmit and receive buffers defined by the Ethernet buffer descriptors. Because Ethernet data buffers are located in RAM outside of the Ethernet controllers, general-purpose DMA engines can move data to or from these memory regions. Also, because Ethernet controllers automatically read buffer descriptors and send (or load) data buffers, a DMA transfer into (or out of) these buffers is effectively a transfer into (or out of) the Ethernet ports.

15.5.1.3 DMA to Configuration and Control Registers

Because any internal register can be addressed with the four-channel DMA controller, configuration and control registers throughout the device are valid DMA targets. However, the primary purpose of DMA—to reduce processor load by moving large blocks of data— is not served by DMA transfers of configuration data. For example, while it is possible to DMA into the 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.

15.5.1.4 DMA to I^2C

The I^2C controller is not transparent to DMA transfers. Observe the caveats listed in Section 15.5.1.3, “DMA to Configuration and Control Registers,” when accessing any I^2C register, including the data register (I2CDR).

15.5.1.5 DMA to DUART

The DUART provides complete and sophisticated DMA support which is described in Chapter 12, “DUART,” specifically Section 12.4.5, “FIFO Mode.”

Chapter 16
PCI/PCI-X Bus Interface

The MPC8540 PCI/PCI-X interface complies with the *PCI Local Bus Specification*, Rev. 2.2 and the *PCI-X Addendum to the PCI Local Bus Specification*, Rev. 1.0a. It is beyond the scope of this manual to document the intricacies of the PCI and PCI-X buses. This chapter 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 MPC8540 implements the PCI and PCI-X buses. Designers of systems incorporating PCI/PCI-X devices should refer to the respective specification for a thorough description of the PCI/PCI-X buses.

NOTE

Much of the available PCI literature refers to a 16-bit quantity as a WORD and a 32-bit quantity as a DWORD. 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.

16.1 Introduction

The PCI/X controller acts as a bridge between the PCI/X interface and the OCeaN switch fabric. Figure 16-1 is a high-level block diagram of the PCI/X controller.

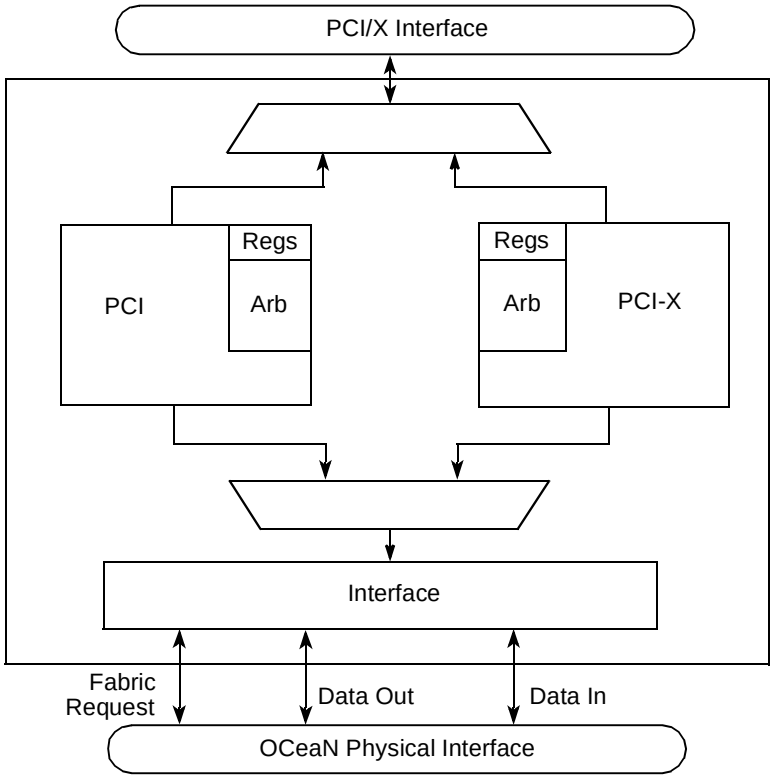


Figure 16-1. PCI/X Controller Block Diagram

16.1.1 Overview

The PCI/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 MPC8540 functions both as a master (initiator) and a target device. Internally, the design is divided into the following:

- Data path blocks
- Control logic blocks
- Memory

The data path blocks contain the queues, tables for transaction tracking, and ordering. The control blocks contain control logic and state-machines for buffer control, bus protocol, tag generation, and transaction resizing. The memory blocks are used solely for inbound and outbound data storage.

This allows the MPC8540 to handle separate PCI/X transactions simultaneously. For example, consider the case where a burst-write transaction from the MPC8540 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 MPC8540, as a target, can accept the burst-read transfer. When the MPC8540 is granted mastership of the PCI bus, the burst-write transaction continues.

There are two blocks of memory in the design:

- The inbound buffers
- The outbound read buffers combined with the outbound write buffers

There are many blocks of control logic in the block. On the PCI/X side there are machines for PCI/X controller-initiated address and data tenures for inbound and outbound data, respectively. On the OCeaN side there are machines for fabric arbitration, outbound data, and inbound data.

As an initiator, the MPC8540 supports read and write operations to the PCI/X memory space, the PCI/X I/O space, and the 256-byte PCI/X configuration space. As an initiator, the MPC8540 also supports generating PCI/X special-cycle and interrupt-acknowledge transactions. As a target, the MPC8540 supports read and write operations to local memory, and, when configured in agent mode, read and write operations to the internal PCI/X configuration registers.

The MPC8540 can function as either a PCI/X host bridge (host mode) or a peripheral device on the PCI/X bus (agent mode). See Section 16.1.3.1.1, “Host Mode,” for more information.

In agent mode, all of the PCI/X configuration registers in the MPC8540 can be programmed from the PCI/X bus. See Section 16.4.2.11.3, “PCI Configuration in Agent and Agent Lock Modes,” for more information.

The PCI/X interface provides bus arbitration for the MPC8540 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 MPC8540 also provides an address translation mechanism to map inbound PCI/X to OCeaN accesses and outbound OCeaN to PCI/X accesses.

16.1.1.1 MPC8540 as a PCI/X Initiator

Upon detecting an OCeaN-to-PCI/X transaction, the MPC8540 requests the use of the PCI/X bus. For OCeaN-to-PCI/X bus write operations, the MPC8540 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 MPC8540 requests mastership of the PCI/X bus when it decodes that the access is for PCI/X address space.

Once granted, the MPC8540 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.

16.1.1.2 MPC8540 as a PCI/X Target

Upon detection of a PCI/X address phase, the MPC8540 decodes the address and bus command to determine if the transaction is within the local memory access boundaries. If the transaction is destined for local memory, the target interface latches the address, decodes the PCI/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 MPC8540 can issue target-abort, target-retry, and target-disconnect cycles. The target interface supports fast back-to-back transactions. The target interface uses the fastest device selection timing. As a PCI/X target, the MPC8540 can issue split response and split completion transactions.

The MPC8540 supports data streaming to and from local memory. This means that data can flow between the MPC8540's PCI/X interface and local memory as long as the internal buffers are not filled.

16.1.2 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

16.1.3 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 16-1 provides a summary of these modes.

Table 16-1. 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.	4.4.3.12/4-19
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].	4.4.3.15/4-20
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

16.1.3.1 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 16.5.1, "Power-On Reset Configuration Modes."

16.1.3.1.1 Host Mode

When the device powers up in host mode, all inbound configuration accesses are ignored (and thus master aborted). See Section 16.5.1.1, “Host Mode,” for more information.

16.1.3.1.2 Agent Mode

When the device powers up in agent mode, it acknowledges inbound configuration accesses. See Section 16.5.1.2, “Agent Mode,” for more information. 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.

16.1.3.1.3 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 16.5.1.3, “Agent Configuration Lock Mode,” for more information.

16.1.3.2 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 *MPC8540 Integrated Processor Hardware Specifications*, for more information.

16.1.3.3 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 *MPC8540 Integrated Processor Hardware Specifications*.

16.1.3.4 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 *MPC8540 Integrated Processor Hardware Specifications*.

16.1.3.5 PCI/X Signal Output Hold Timing

To meet minimum output hold specifications relative to SYSCLK for both PCI and PCI-X systems, the MPC8540 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 *MPC8540 Integrated Processor Hardware Specifications*, for more information on these values and signal timing.

16.1.3.6 PCI/X Impedance

The MPC8540 has a programmable impedance for PCI/X bus signals. The initial value is determined by the value on the $\overline{\text{PCI\_GNT1}}$ power-on reset configuration signal. See Chapter 4, “Reset, Clocking, and Initialization,” and the *MPC8540 Integrated Processor Hardware Specifications*, for more information.

16.2 External Signal Descriptions

Figure 16-2 shows the external PCI/X signals.

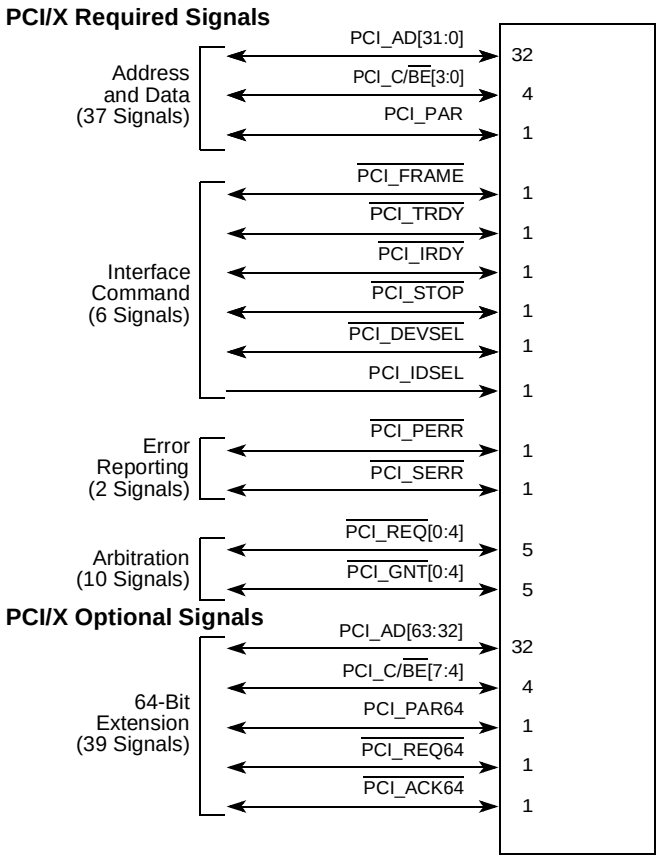


Figure 16-2. PCI/X Interface External Signals

Table 16-2 contains the detailed descriptions of the external PCI/X interface signals.

Table 16-2. 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 16-2. PCI Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
PCI_C/$\overline{BE}$[7:0]	I/O	Command/byte enable. Command encodings are described in Section 16.4.2.2, "PCI Bus Commands," and Section 16.4.3.2, "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/$\overline{BE}$[7:0] define the bus command. Byte enables determine which byte lanes carry meaningful data for PCI bus data phases. The PCI_C/$\overline{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{BE}$[7:0] indicate the command that another master is sending. During the PCI bus data phase, PCI_C/$\overline{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
$\overline{PCI\_DEVSEL}$	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 16-2. 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 MPC8540 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 16-2. 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 16-2. 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 MPC8540 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 16-2. 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 MPC8540 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 16-2. 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

16.3 Memory Map/Register Definitions

The PCI/X interface unit of the MPC8540 supports the following register types:

- Memory-mapped registers—these registers control PCI address translation, PCI error management, and PCI configuration register access on the MPC8540. These registers are described in Section 16.3.1, "PCI/X Memory Mapped Registers," and its subsections.
- PCI/X configuration registers contained within the PCI/X configuration header—these registers are specified by the PCI bus specification for every PCI device. These registers are described in Section 16.3.2, "PCI/X Configuration Header," and its subsections.

16.3.1 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 16-3 lists the memory-mapped registers.

Table 16-3. 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	16.3.1.1.1/16-18
0x0_8004	CFG_DATA—PCI/X configuration data	R/W	0x0000_0000	16.3.1.1.1/16-18
0x0_8008	INT_ACK—PCI/X interrupt acknowledge	R	0x0000_0000	16.3.1.1.3/16-20
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	16.3.1.2.1/16-21
0x0_8C04	POTEAR0—PCI/X outbound window 0 (default) translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-21
0x0_8C08	Reserved	—	—	
0x0_8C0C	Reserved	—	—	
0x0_8C10	POWAR0—PCI/X outbound window 0 (default) attributes register	R/W	0x8004_401F	16.3.1.2.4/16-22
0x0_8C14–0x0_8C1C	Reserved	—	—	

Table 16-3. 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	16.3.1.2.1/16-21
0x0_8C24	POTEAR1—PCI/X outbound window 1 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-21
0x0_8C28	POWBAR1—PCI/X outbound window 1 base address register	R/W	0x0000_0000	16.3.1.2.3/16-22
0x0_8C2C	Reserved	—	—	
0x0_8C30	POWAR1—PCI/X outbound window 1 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
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	16.3.1.2.1/16-21
0x0_8C44	POTEAR2—PCI/X outbound window 2 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-21
0x0_8C48	POWBAR2—PCI/X outbound window 2 base address register	R/W	0x0000_0000	16.3.1.2.3/16-22
0x0_8C4C	Reserved	—	—	
0x0_8C50	POWAR2—PCI/X outbound window 2 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
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	16.3.1.2.1/16-21
0x0_8C64	POTEAR3—PCI/X outbound window 3 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-21
0x0_8C68	POWBAR3—PCI/X outbound window 3 base address register	R/W	0x0000_0000	16.3.1.2.3/16-22
0x0_8C6C	Reserved	—	—	
0x0_8C70	POWAR3—PCI/X outbound window 3 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
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	16.3.1.2.1/16-21
0x0_8C84	POTEAR4—PCI/X outbound window 4 translation extended address register	R/W	0x0000_0000	16.3.1.2.2/16-21

Table 16-3. 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	16.3.1.2.3/16-22
0x0_8C8C	Reserved	—	—	
0x0_8C90	POWAR4—PCI/X outbound window 4 attributes register	R/W	0x0000_0000	16.3.1.2.4/16-22
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	16.3.1.3.1/16-25
0x0_8DA4	Reserved	—	—	
0x0_8DA8	PIWBAR3—PCI/X inbound window 3 base address register	R/W	0x0000_0000	16.3.1.3.2/16-25
0x0_8DAC	PIWBEAR3—PCI/X inbound window 3 base extended address register	R/W	0x0000_0000	16.3.1.3.3/16-26
0x0_8DB0	PIWAR3—PCI/X inbound window 3 attributes register	R/W	0x0000_0000	16.3.1.3.4/16-26
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	16.3.1.3.1/16-25
0x0_8DC4	Reserved	—	—	
0x0_8DC8	PIWBAR2—PCI/X inbound window 2 base address register	R/W	0x0000_0000	16.3.1.3.2/16-25
0x0_8DCC	PIWBEAR2—PCI/X inbound window 2 base extended address register	R/W	0x0000_0000	16.3.1.3.3/16-26
0x0_8DD0	PIWAR2—PCI/X inbound window 2 attributes register	R/W	0x0000_0000	16.3.1.3.4/16-26
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	16.3.1.3.1/16-25
0x0_8DE4	Reserved	—	—	
0x0_8DE8	PIWBAR1—PCI/X inbound window 1 base address register	R/W	0x0000_0000	16.3.1.3.2/16-25
0x0_8DEC	Reserved	—	—	
0x0_8DF0	PIWAR1—PCI/X inbound window 1 attributes register	R/W	0x0000_0000	16.3.1.3.4/16-26
0x0_8DF4–0x0_8DFC	Reserved	—	—	
PCI/X Error Management Registers				

Table 16-3. PCI/X Memory-Mapped Register Map (continued)

Offset	Register	Access	Reset	Section/Page
0x0_8E00	ERR_DR—PCI/X error detect register	Special	0x0000_0000	16.3.1.4.1/16-29
0x0_8E04	ERR_CAP_DR—PCI/X error capture disabled register	R/W	0x0000_0000	16.3.1.4.2/16-30
0x0_8E08	ERR_EN—PCI/X error enable register	R/W	0x0000_0000	16.3.1.4.3/16-31
0x0_8E0C	ERR_ATTRIB—PCI/X error attributes capture register	R/W	0x0000_0000	16.3.1.4.4/16-32
0x0_8E10	ERR_ADDR—PCI/X error address capture register	R/W	0x0000_0000	16.3.1.4.5/16-33
0x0_8E14	ERR_EXT_ADDR—PCI/X error extended address capture register	R/W	0x0000_0000	16.3.1.4.6/16-34
0x0_8E18	ERR_DL—PCI/X error data low capture register	R/W	0x0000_0000	16.3.1.4.7/16-34
0x0_8E1C	ERR_DH—PCI/X error data high capture register	R/W	0x0000_0000	16.3.1.4.8/16-34
0x0_8E20	GAS_TIMR—PCI/X gasket timer register	R/W	0x0000_0000	16.3.1.4.9/16-35
0x0_8E24	PCIX_TIMR—PCI-X split completion timer register	R/W	0x0000_0000	16.3.1.4.10/16-36
0x0_8E28–0x0_8EFC	Reserved	—	—	
0x0_8F00–0x0_8FFC	Reserved for debug	—	—	

16.3.1.1 PCI/X Configuration Access Registers

The PCI/X configuration header, shown in Figure 16-25 and Figure 16-26, 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.

16.3.1.1.1 PCI/X Configuration Address Register (CFG_ADDR)

The CFG_ADDR register is shown in Figure 16-3.

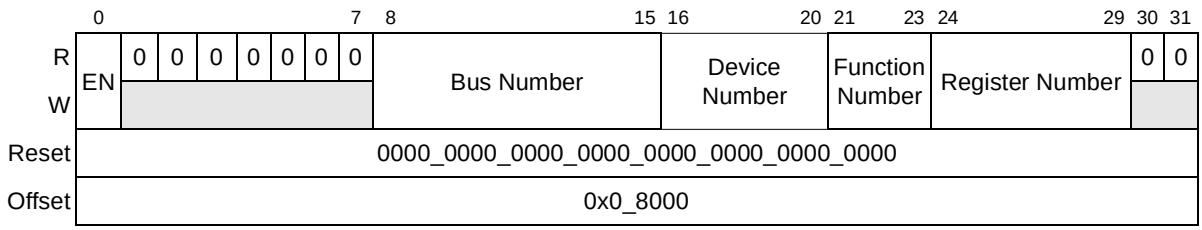


Figure 16-3. PCI/X CFG_ADDR Register

Table 16-4 describes the bit settings for the CFG_ADDR register.

Table 16-4. 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 MPC8540 itself.

See Section 16.4.2.11.2, “Accessing the PCI Configuration Space in Host Mode,” and Section 16.4.2.11.3, “PCI Configuration in Agent and Agent Lock Modes,” for usage of PCI/X CFG_ADDR.

16.3.1.1.2 PCI/X Configuration Data Register (CFG_DATA)

The CFG_DATA register is shown in Figure 16-3.

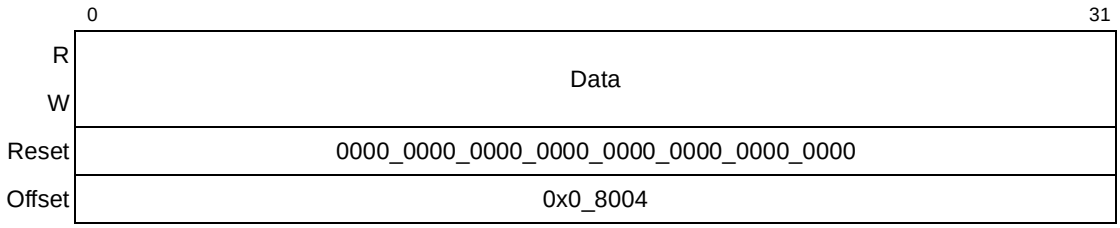


Figure 16-4. PCI/X CFG_DATA Register

Table 16-5 describes the bit settings for the CFG_DATA register

Table 16-5. 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 16.4.2.11.2, “Accessing the PCI Configuration Space in Host Mode,” and Section 16.4.2.11.3, “PCI Configuration in Agent and Agent Lock Modes,” for use of CFG_DATA.

16.3.1.1.3 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 16-5.

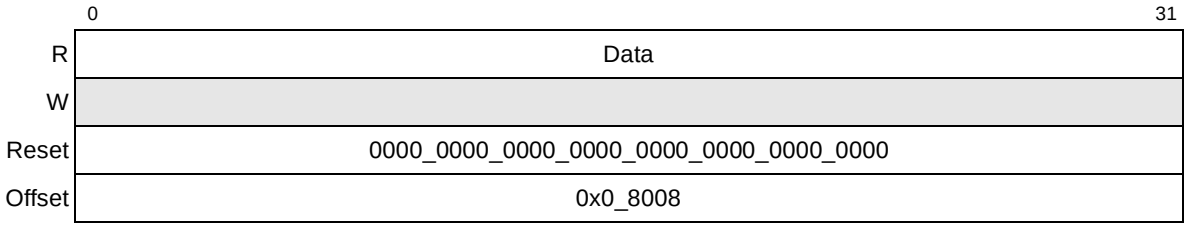


Figure 16-5. PCI/X INT_ACK Register

Table 16-6 describes the bit settings for the INT_ACK register.

Table 16-6. PCI/X INT_ACK Field Descriptions

Bits	Name	Description
0–31	Data	A read to this register generates a PCI interrupt acknowledge cycle.

16.3.1.2 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 MPC8540 to the external PCI address space. The outbound ATMU consists of four translation windows plus a default translation for transactions that do not hit in one of the four windows.

Each window contains a base address that points to the beginning of the window in the local address map, a translation address that specifies the high-order bits of the transaction in the external PCI address space, and a set of attributes including window size and external transaction type.

Each window must be aligned 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.

16.3.1.2.1 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 16-6.

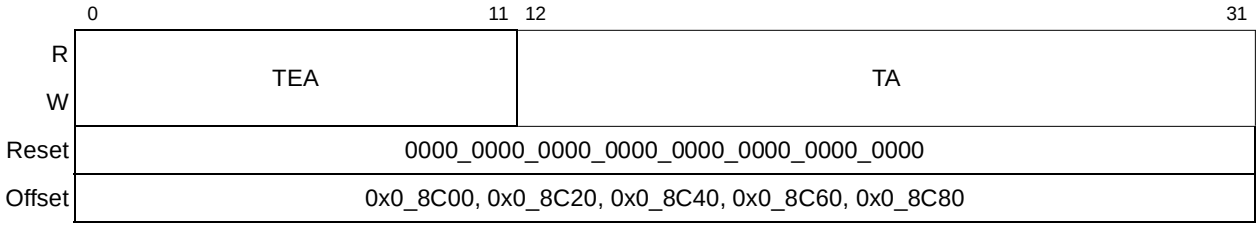


Figure 16-6. PCI/X Outbound Translation Address Registers

Table 16-7 describes the fields of the POTAR*n* registers.

Table 16-7. 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.

16.3.1.2.2 PCI/X Outbound Translation Extended Address Registers (POTEAR*n*)

The PCI/X outbound translation extended address registers (POTEAR*n*), shown in Figure 16-7, contain the most significant bits of a 64-bit translation address.

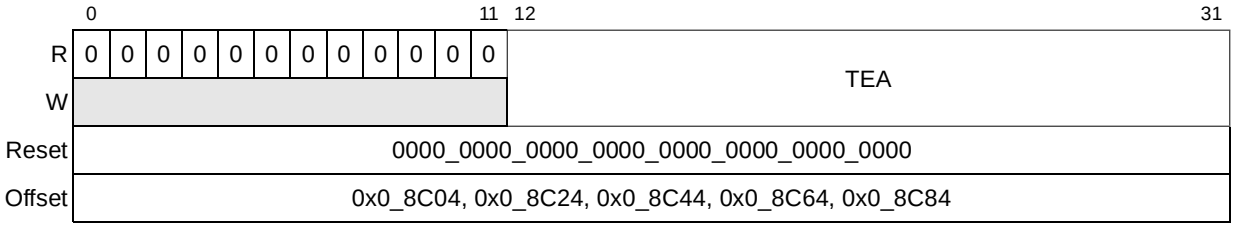


Figure 16-7. PCI/X Outbound Translation Extended Address Registers

Table 16-8 describes the fields of the POTEAR*n*.

Table 16-8. 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*)

16.3.1.2.3 PCI/X Outbound Window Base Address Registers (POWBAR*n*)

The PCI/X outbound window base address registers (POWBAR*n*), shown in Figure 16-8, 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.

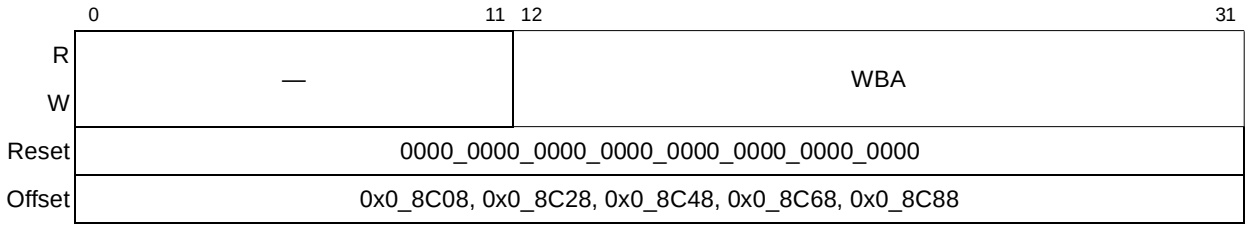


Figure 16-8. PCI/X Outbound Window Base Address Registers

Table 16-9 describes the field of the POWBAR*n*.

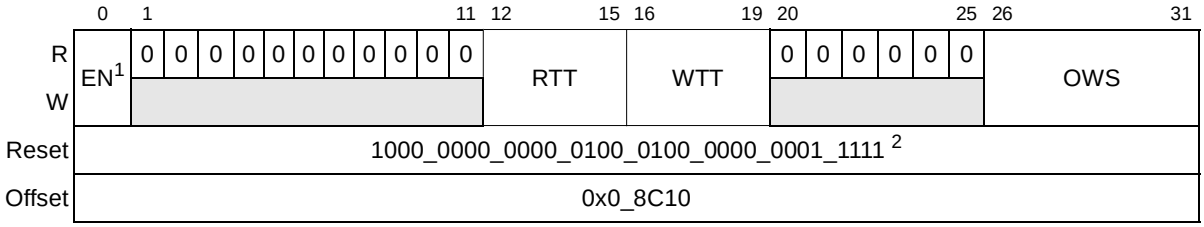
Table 16-9. POWBAR*n* Field Descriptions

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.

16.3.1.2.4 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 16-9. 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 16-9. PCI/X Outbound Window Attributes Register 0 (Default)

POWAR1–POWAR4 are shown in Figure 16-10.

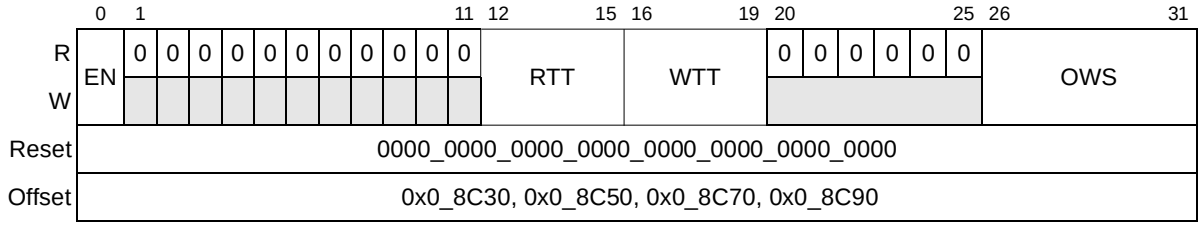


Figure 16-10. PCI/X Outbound Window Attributes Registers 1–4

Table 16-10 describes the fields for the POWAR*n* registers.

Table 16-10. 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 0000–0011Reserved 0100 Memory Read 0101–0111Reserved 1000 I/O Read 1001–1111Reserved
16–19	WTT	Writetransaction type to run on PCI 0000–0011Reserved 0100 Memory Write 0101–0111Reserved 1000 I/O Write 1001–1111Reserved

Table 16-10. 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. 000000–0010114 Kbyte window size 0011008-Kbyte window size ... 0111114-Gbyte window size 100000–111111Reserved 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.

16.3.1.3 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 MPC8540. 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 16-25. 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 16.3.1.1.1, "PCI/X Configuration Address Register (CFG_ADDR)," Section 16.3.1.1.2, "PCI/X Configuration Data Register (CFG_DATA)," and Section 16.3.2.11, "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.

16.3.1.3.1 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 16-11.

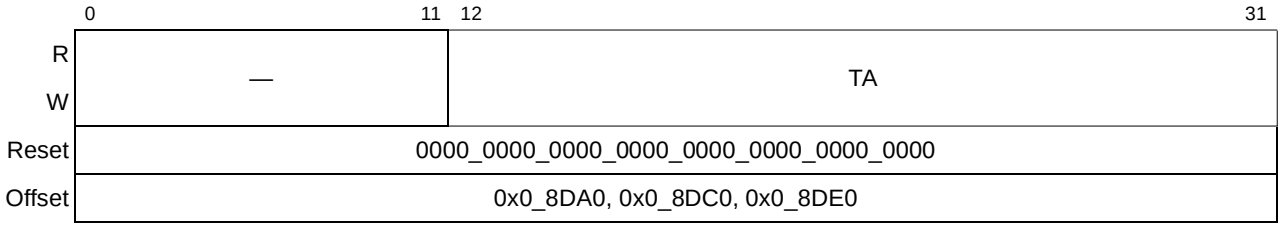


Figure 16-11. PCI/X Inbound Translation Address Registers

Table 16-11 describes the fields of the PITAR*n* registers.

Table 16-11. 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.

16.3.1.3.2 PCI/X Inbound Window Base Address Registers (PIWBAR*n*)

PCI/X inbound window base address registers (PIWBAR*n*), shown in Figure 16-12, select the PCI/X base address for windows translated to the MPC8540 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.

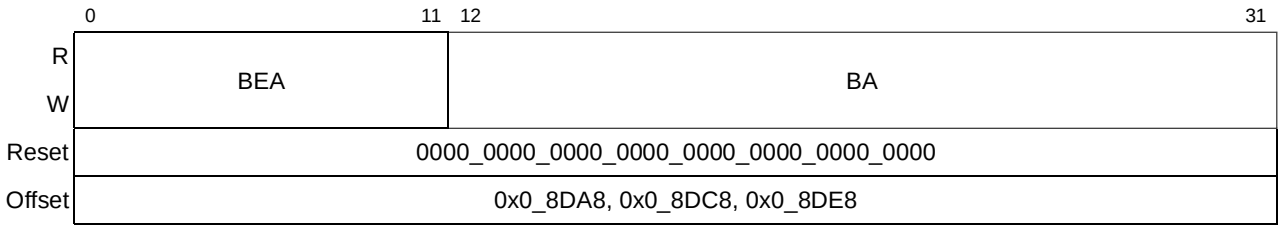


Figure 16-12. PCI/X Inbound Window Base Address Registers

Table 16-12 describes the fields of the PIWBAR*n* registers.

Table 16-12. 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.

16.3.1.3.3 PCI/X Inbound Window Base Extended Address Registers (PIWBEAR*n*)

The PCI/X inbound window base extended address registers (PIWBEAR*n*), shown in Figure 16-13, 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.

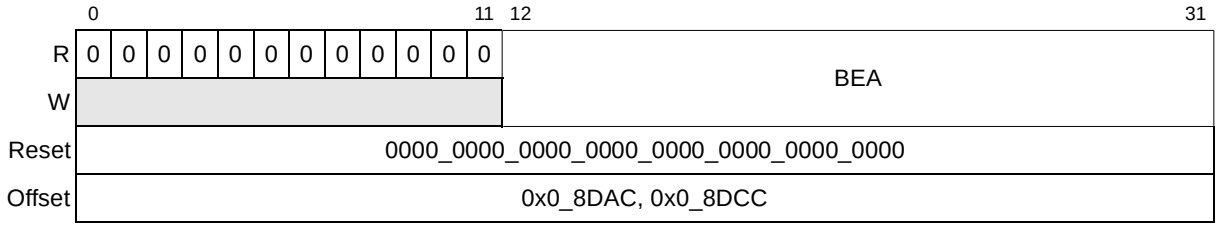


Figure 16-13. PCI/X Inbound Window Base Extended Address Registers

Table 16-13 describes the fields of the PIWBEAR*n* registers.

Table 16-13. 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.

16.3.1.3.4 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 16-14.

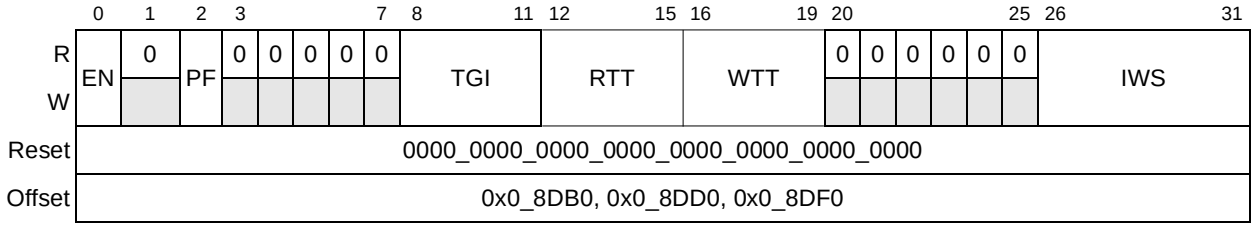


Figure 16-14. PCI/X Inbound Window Attributes Registers

Table 16-14 describes the fields of the PIWAR*n* registers.

Table 16-14. 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 16-14. 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.

16.3.1.4 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 16-57 for PCI mode error actions and Table 16-65 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 MPC8540 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 MPC8540 does drive the correct parity for inbound read transactions that it terminates with a split response.

See Section 16.4.2.13, “PCI Error Functions,” and Section 16.4.3.8, “PCI-X Error Functions,” for more details on error handling.

16.3.1.4.1 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 16-15. PCI/X Error Detect Register (ERR_DR)

Table 16-15 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 16-15. 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)

16.3.1.4.2 PCI/X Error Capture Disable Register (ERR_CAP_DR)

	0–20	21	22	23	24	25	26	27	28	29	30	31
R W		Addr parity error capture disable	$\overline{\text{Rcvd}}$ $\overline{\text{SERR}}$ error capture disable	$\overline{\text{Mstr}}$ $\overline{\text{PERR}}$ error capture disable	$\overline{\text{Trgt}}$ $\overline{\text{PERR}}$ error capture disable	Mstr abort error capture disable	Trgt abort error capture disable	OWMSV error capture disable	ORMSV error capture disable	IRMSV error capture disable	SCM error capture disable	TOE error capture disable
Reset	0000_0000_0000_0000_0000_0000_0000_0000											
Offset	0x0_8E04											

Figure 16-16. PCI/X Error Capture Disable Register (ERR_CAP_DR)

Table 16-16. 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

16.3.1.4.3 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 16-17. PCI/X Error Enable Register (ERR_EN)

Table 16-17 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 16-17. 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

16.3.1.4.4 PCI/X Error Attributes Capture Register (ERR_ATTRIB)

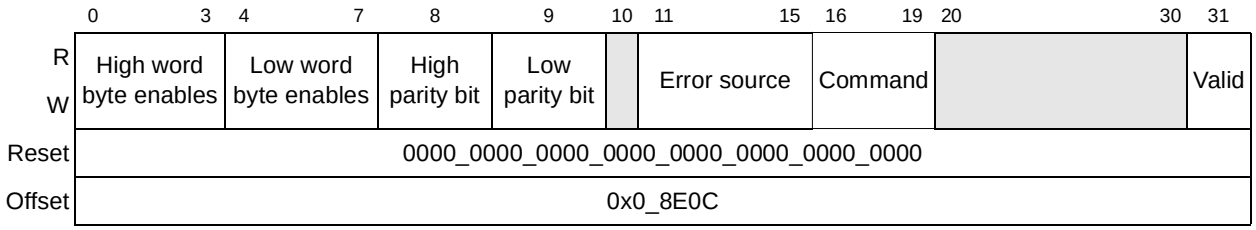


Figure 16-18. PCI/X Error Attributes Capture Register (ERR_ATTRIB)

Table 16-18. 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 16-18. 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 Reserved 10101 DMA 10110 RDC 10111 SAP 11000 TSEC1 11001 TSEC2 11010 FEC (10/100) 11011 Reserved 11100 RapidIO message units 11101 RapidIO doorbell units 11110 RapidIO port-write units 11111 Reserved
16–19	Command	PCI command
20–30	—	Reserved
31	Valid info	The PCI bus capture registers contain valid information

16.3.1.4.5 PCI/X Error Address Capture Register (ERR_ADDR)

	0	31
R W	Memory Address	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
Offset	0x0_8E10	

Figure 16-19. PCI/X Error Address Capture Register (ERR_ADDR)

Table 16-19. ERR_ADDR Field Descriptions

Bits	Name	Description
0–31	Memory address	Memory transaction address

16.3.1.4.6 PCI/X Error Extended Address Capture Register (ERR_EXT_ADDR)

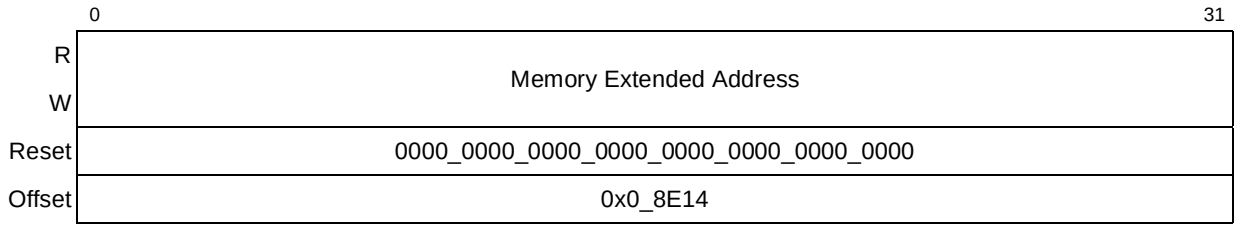


Figure 16-20. PCI/X Error Extended Address Capture Register (ERR_EXT_ADDR)

Table 16-20. ERR_EXT_ADDR Field Descriptions

Bits	Name	Description
0–31	Memory extended address	Memory transaction extended address

16.3.1.4.7 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).

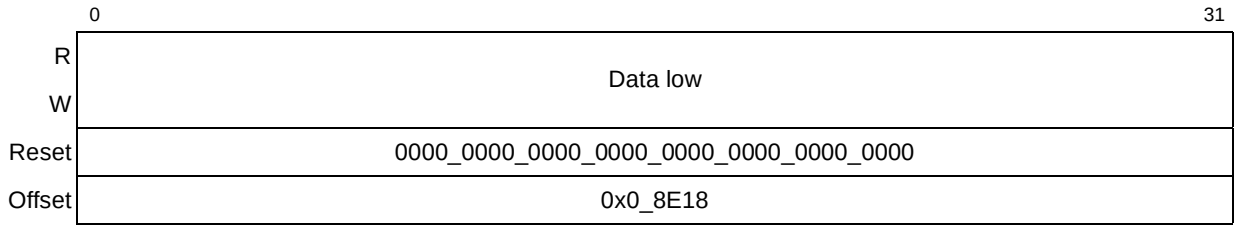


Figure 16-21. PCI/X Error Data Low Capture Register (ERR_DL)

Table 16-21. ERR_DL Field Description

Bits	Name	Description
0–31	Data low	Least significant PCI bus data word

16.3.1.4.8 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.

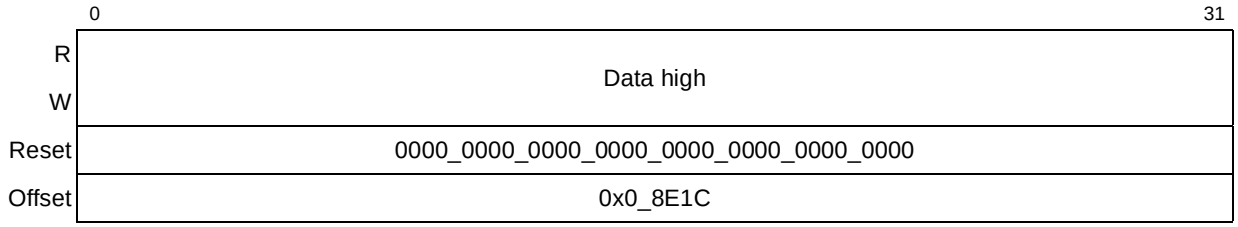


Figure 16-22. PCI/X Error Data High Capture Register (ERR_DH)

Table 16-22. ERR_DH Field Description

Bits	Name	Description
0–31	Data high	Most significant PCI bus data word (only valid with 64-bit PCI bus)

16.3.1.4.9 PCI/X Gasket Timer Register (GAS_TIMR)

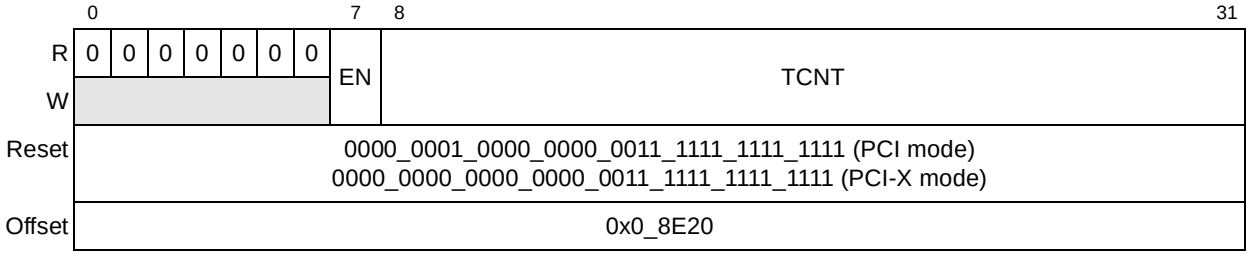


Figure 16-23. PCI/X Gasket Timer Register (GAS_TIMR)

Table 16-23. 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.
8–31	TCNT	Number of system clocks to purge a non-prefetchable inbound read buffer

16.3.1.4.10 PCI-X Split Completion Timer Register (PCIX_TIMR)

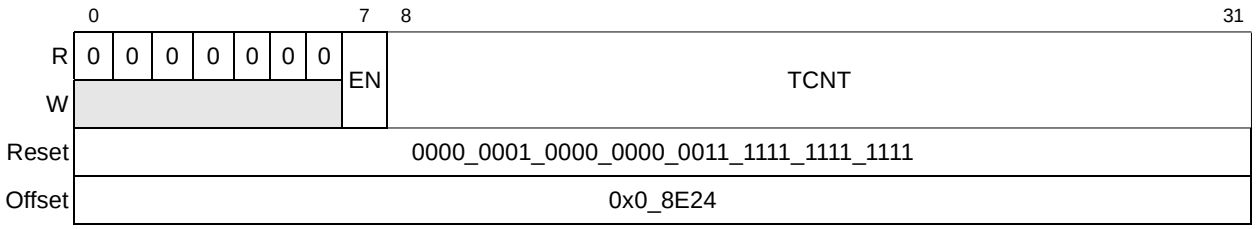


Figure 16-24. PCI-X Split Completion Timer Register (PCIX_TIMR)

Table 16-24. 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.

16.3.2 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 16-25 lists the common PCI/X configuration header as implemented by the MPC8540.

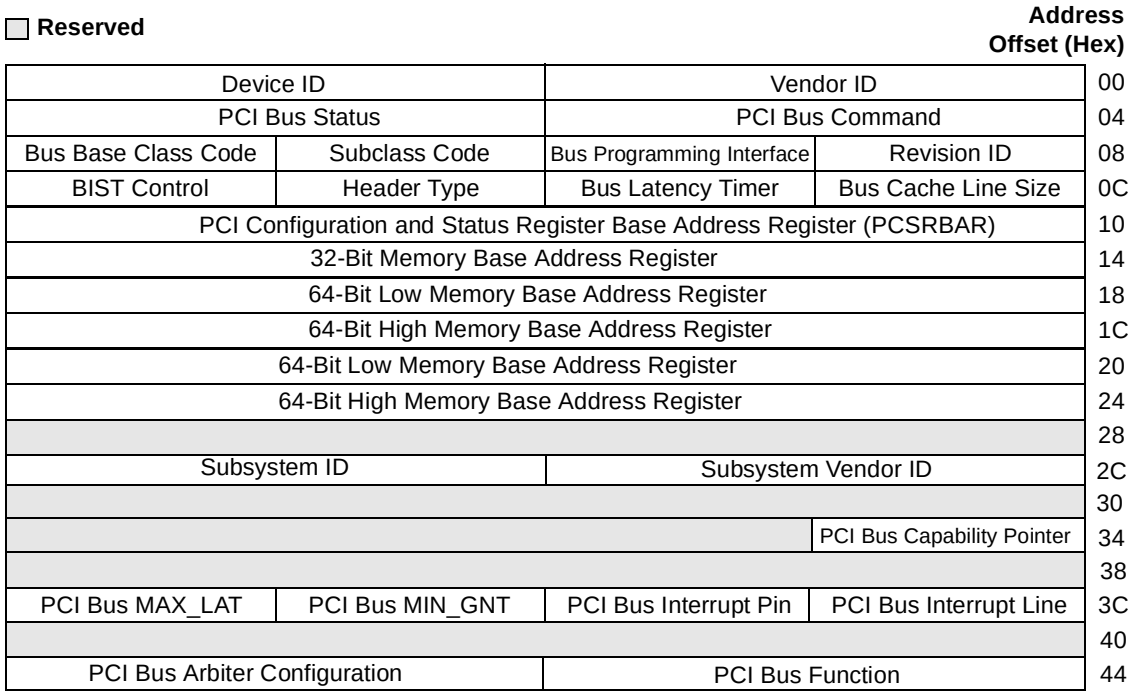


Figure 16-25. MPC8540 PCI/X Configuration Header

Figure 16-26 lists the additional PCI-X configuration registers.

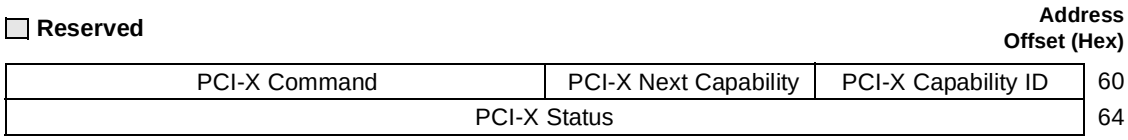


Figure 16-26. PCI-X Additional Configuration Registers

Note that software must be restricted from accessing any reserved or undefined register space on the MPC8540; doing so may hang the device.

Table 16-54 in Section 16.4.2.11.1, "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*.

16.3.2.1 PCI Vendor ID Register—Offset 0x00

The PCI vendor ID register, shown in Figure 16-27, is used to identify the manufacturer of the part.

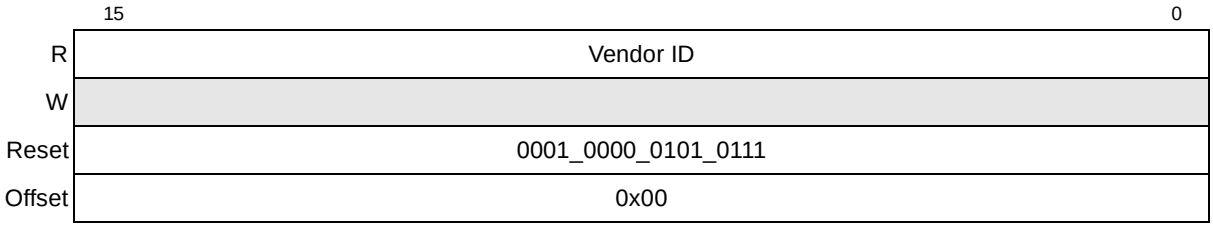


Figure 16-27. PCI Vendor ID Register

Table 16-25 describes PCI vendor ID register fields.

Table 16-25. PCI Vendor ID Register Field Description

Bits	Name	Description
15–0	Vendor ID	0x1057 (Freescale Semiconductor)

16.3.2.2 PCI Device ID Register—Offset 0x02

The PCI device ID register, shown in Figure 16-28, is used to identify the device.

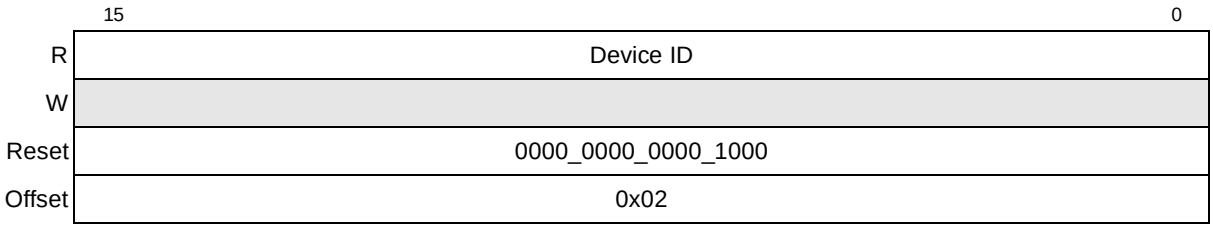


Figure 16-28. PCI Device ID Register

Table 16-26. PCI Device ID Register Field Description

Bits	Name	Description
15–0	Device ID	0x0008

16.3.2.3 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 16-27 describes the bits of the PCI bus command register.

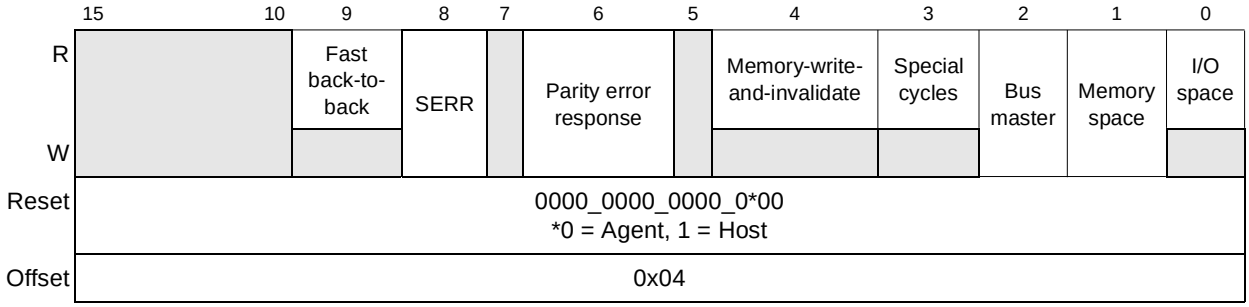


Figure 16-29. PCI Bus Command Register

Table 16-27. 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 MPC8540'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

16.3.2.4 PCI Bus Status Register—Offset 0x06

The 2-byte PCI bus status register is used to record status information for PCI bus bus-related events. The definition of each bit is given in Table 16-28. 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 16-57 for PCI mode error actions and Table 16-65 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.

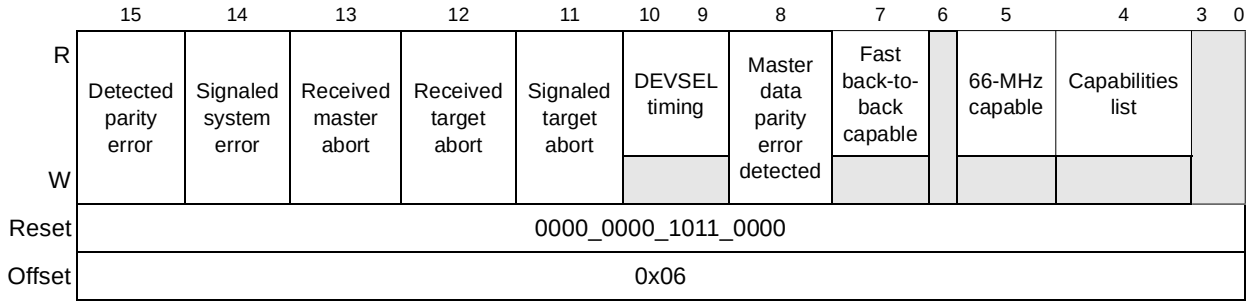


Figure 16-30. PCI Bus Status Register

Table 16-28. 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 16-28. 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

16.3.2.5 PCI Revision ID Register—Offset 0x08

The PCI Revision ID register is used to identify the revision of the part.

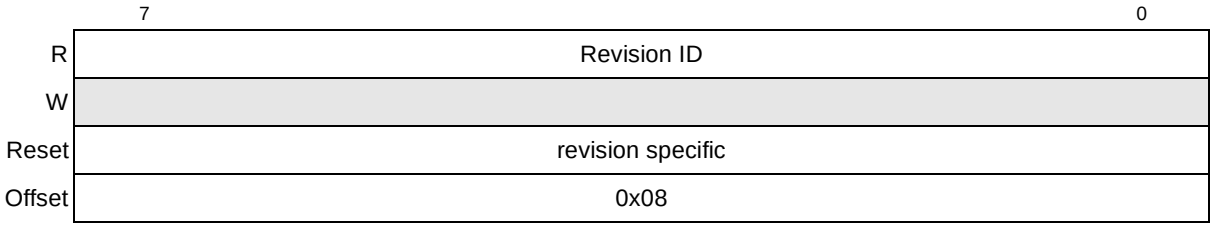


Figure 16-31. PCI Revision ID Register

Table 16-29. PCI Revision ID Register Field Descriptions

Bits	Name	Description
7–0	Revision ID	Revision specific

16.3.2.6 PCI Bus Programming Interface Register—Offset 0x09

Table 16-30 describes the PCI Bus programming interface register (PIR).

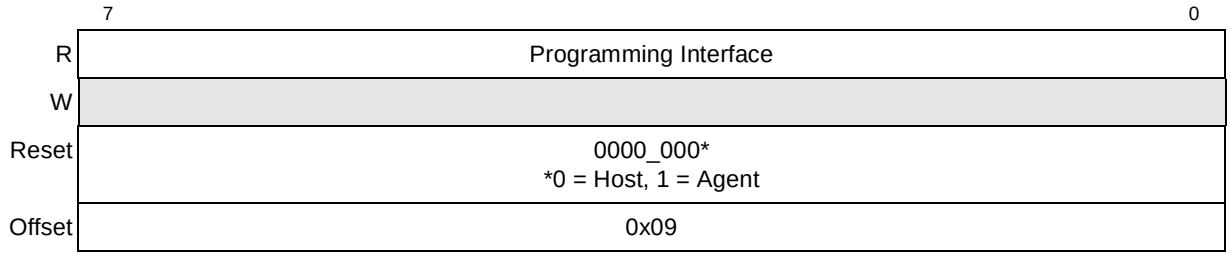


Figure 16-32. PCI Bus Programming Interface Register

Table 16-30. 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

16.3.2.7 PCI Subclass Code Register—Offset 0x0A

Table 16-32 describes the PCI subclass code register (PSCR).

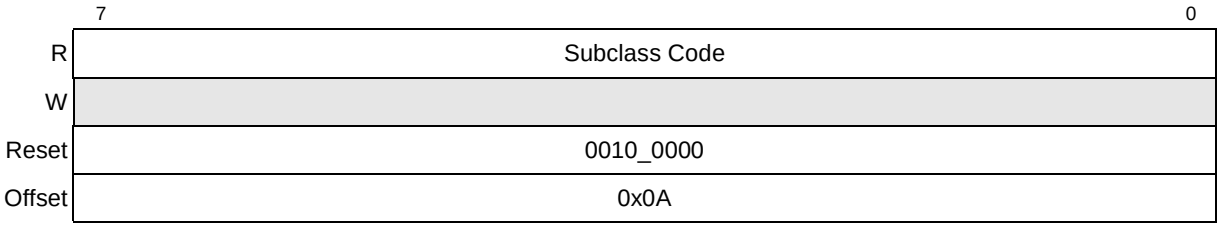


Figure 16-33. PCI Subclass Code Register

Table 16-31. PCI Subclass Code Register Field Description

Bits	Name	Description
7–0	Subclass Code	PowerPC—0x20

16.3.2.8 PCI Bus Base Class Code Register—Offset 0x0B

Table 16-32 describes the PCI bus base class code register (PBCCR).

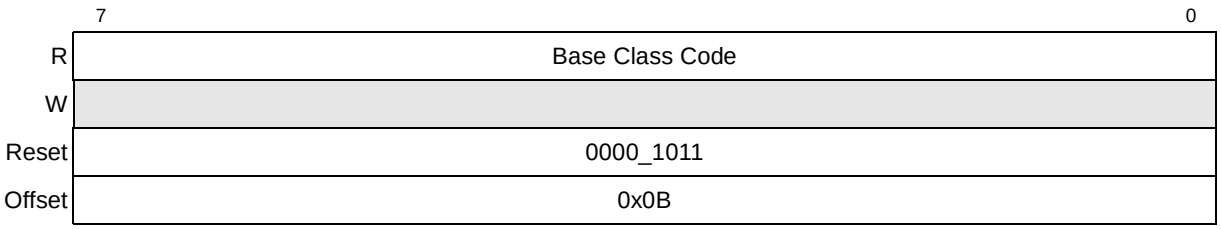


Figure 16-34. PCI Bus Base Class Code Register

Table 16-32. PCI Bus Base Class Code Register Field Description

Bits	Name	Description
7–0	Base Class Code	Processor—0x0B

16.3.2.9 PCI Bus Cache Line Size Register—Offset 0x0C

Table 16-33 describes the PCI bus cache line size register (PCLSR).

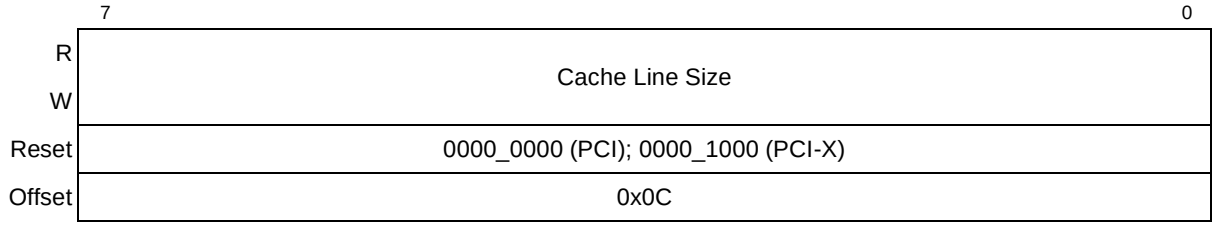


Figure 16-35. PCI Bus Cache Line Size Register (PCLSR)

Table 16-33. 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.

16.3.2.10 PCI Bus Latency Timer Register—0x0D

Table 16-34 describes the PCI latency timer register (PLTR).

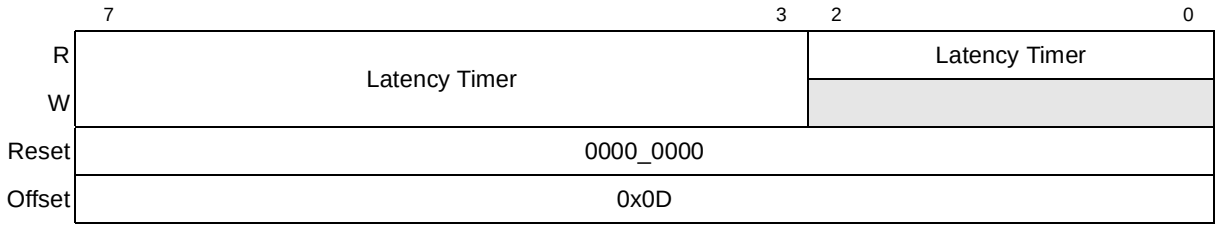


Figure 16-36. PCI Bus Latency Timer Register

Table 16-34. 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.

16.3.2.11 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 16.3.1.3, “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.

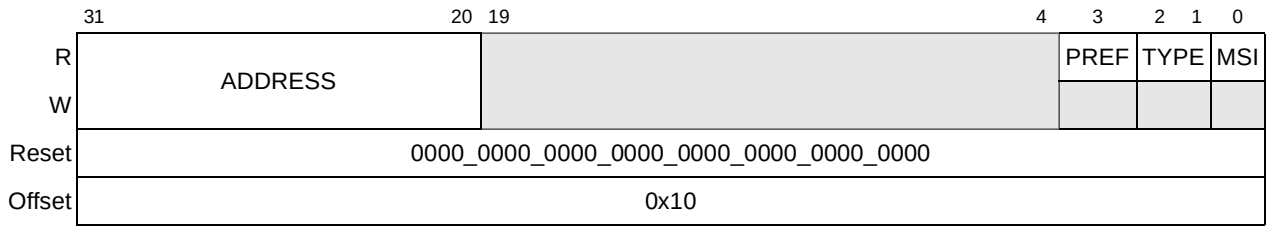


Figure 16-37. PCI Configuration and Status Register Base Address Register (PCSRBAR)

Table 16-35. 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

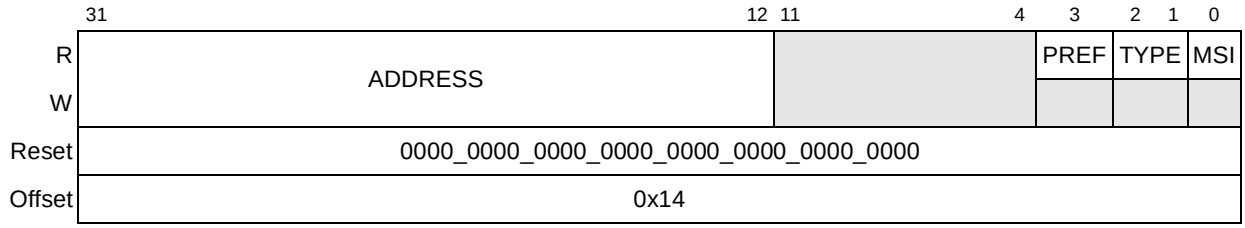


Figure 16-38. 32–Bit Memory Base Address Register

Table 16-36. 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.

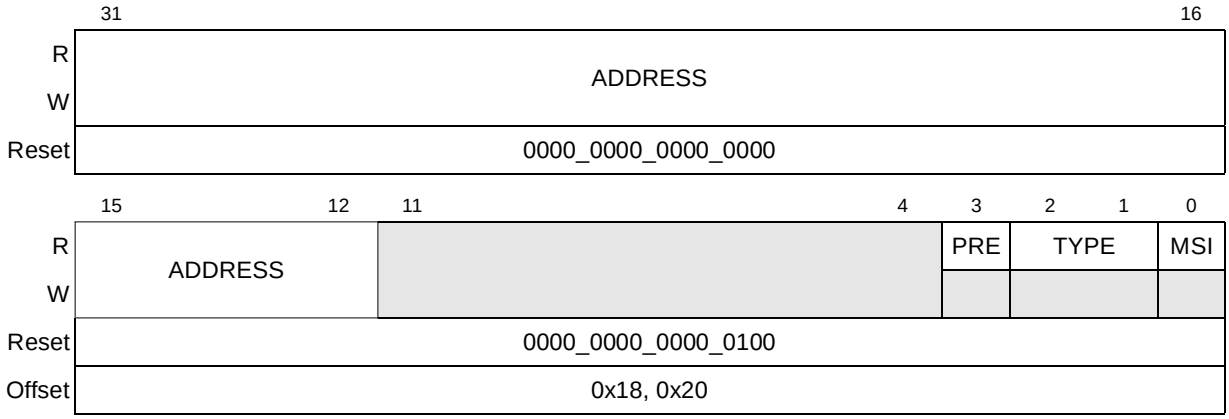


Figure 16-39. 64-Bit Low Memory Base Address Register

Table 16-37. 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

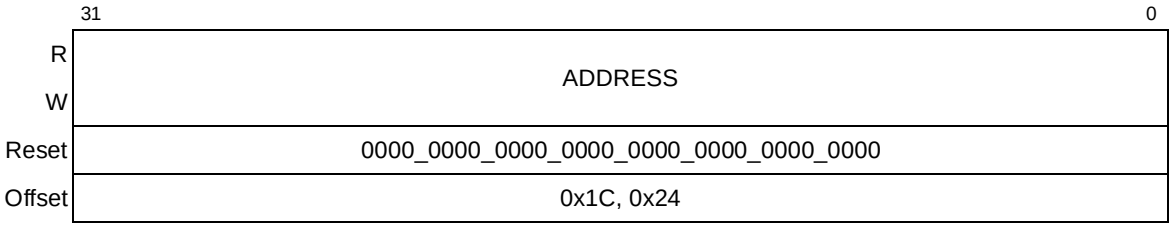


Figure 16-40. 64-Bit High Memory Base Address Register

Table 16-38. 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.

16.3.2.12 PCI Subsystem Vendor ID Register

The PCI subsystem vendor ID register is used to identify the subsystem.

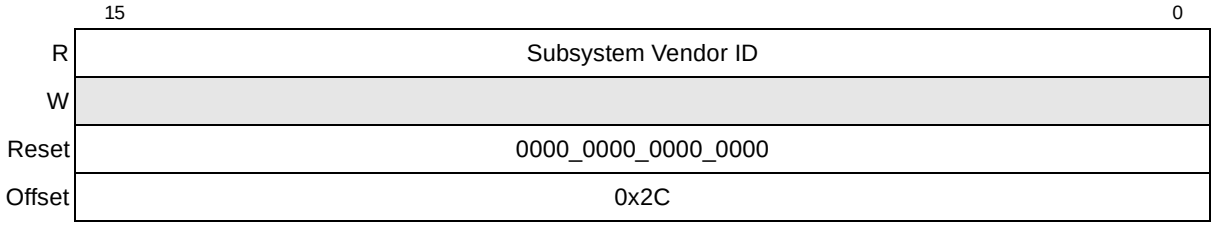


Figure 16-41. PCI Subsystem Vendor ID Register

Table 16-39. PCI Subsystem Vendor ID Register Field Description

Bits	Name	Description
15–0	Subsystem Vendor ID	0x0000

16.3.2.13 PCI Subsystem ID Register

The PCI Subsystem ID register is used to identify the subsystem.

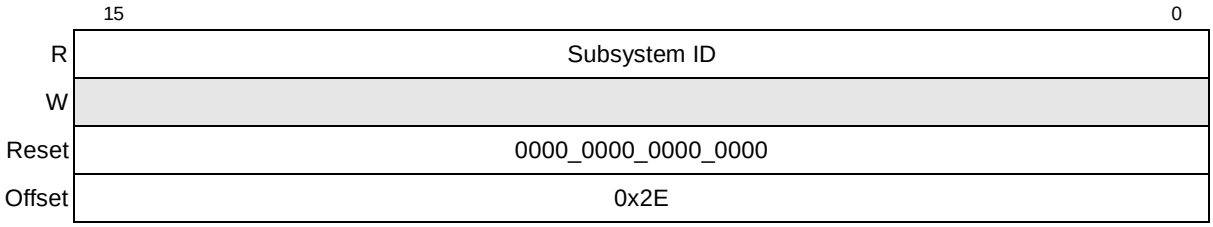


Figure 16-42. PCI Subsystem ID Register

Table 16-40. PCI Subsystem ID Register Field Description

Bits	Name	Description
15–0	Subsystem ID	0x0000

16.3.2.14 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 16-41.

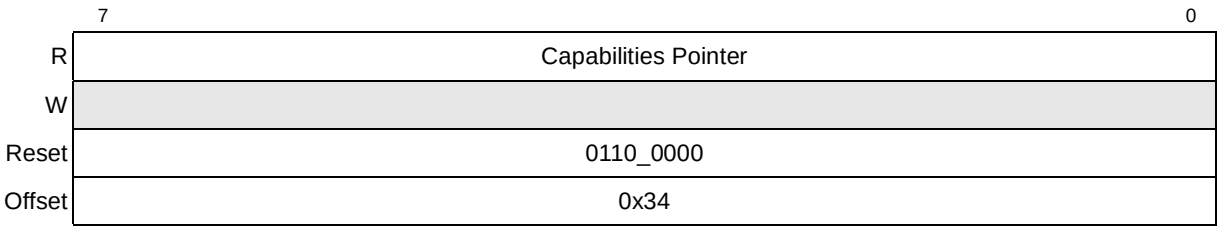


Figure 16-43. PCI Bus Capabilities Pointer Register

Table 16-41. 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.

16.3.2.15 PCI Bus Interrupt Line Register

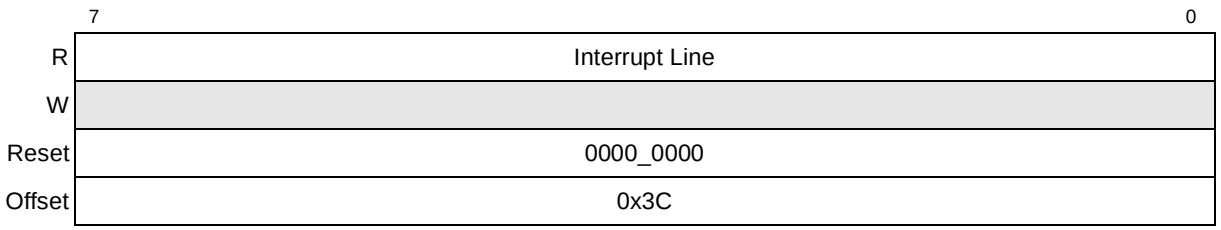


Figure 16-44. PCI Bus Interrupt Line Register

Table 16-42. 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).

16.3.2.16 PCI Bus Interrupt Pin Register

The MPC8540 does not generate specific PCI_INTA, PCI_INTB, etc. outputs, nor does this device respond to INTACK or special cycle commands on the PCI/X interfaces. The MPC8540 does have 12 general purpose interrupt request lines (IRQ[0:11]) and an interrupt output, $\overline{\text{IRQ\_OUT}}$ (active low, level sensitive), to which all external and most internal interrupt sources (including PCI/X) can be routed.

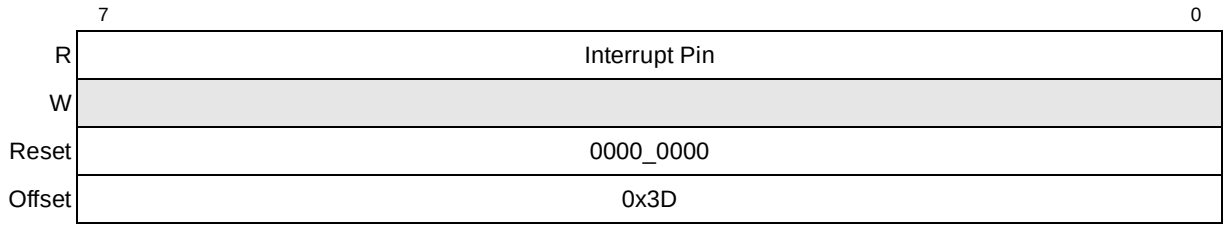


Figure 16-45. PCI Bus Interrupt Pin Register

Table 16-43. PCI Bus Interrupt Pin Register Field Description

Bits	Name	Description
7–0	Interrupt Pin	No PCI_INT pin selected

16.3.2.17 PCI Bus Minimum Grant (MIN_GNT) Register

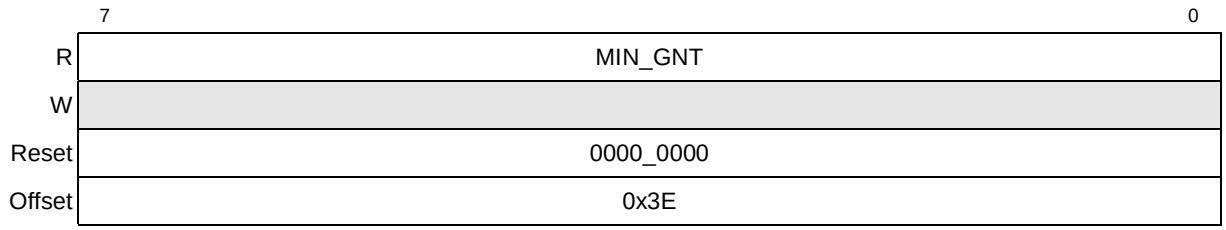


Figure 16-46. PCI Bus Minimum Grant Register

Table 16-44. 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)

16.3.2.18 PCI Bus Maximum Latency (MAX_LAT) Register

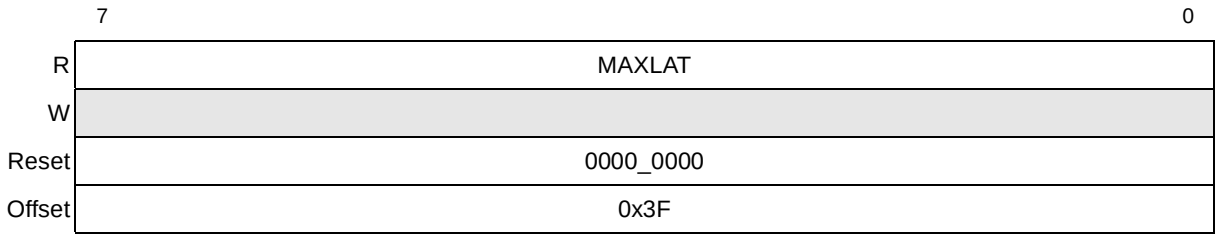


Figure 16-47. PCI Bus Maximum Latency Register

Table 16-45. 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)

16.3.2.19 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.

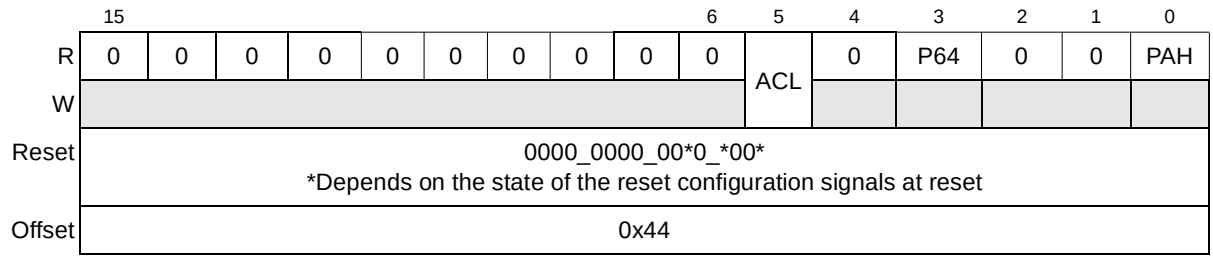


Figure 16-48. PCI Bus Function Register

Table 16-46. 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 16-46. 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

16.3.2.20 PCI Bus Arbiter Configuration Register (PBACR)

The PCI bus arbiter configuration register is used to determine the configuration of the PCI bus arbiter.

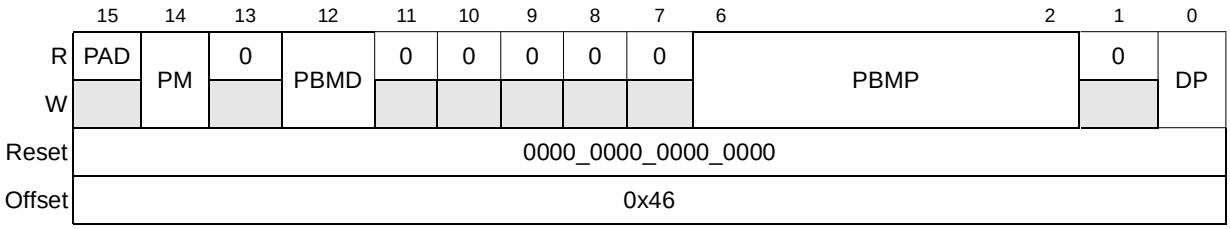


Figure 16-49. PCI Bus Arbiter Configuration Register

Table 16-47. 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 16-47. 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.

16.3.2.21 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*.

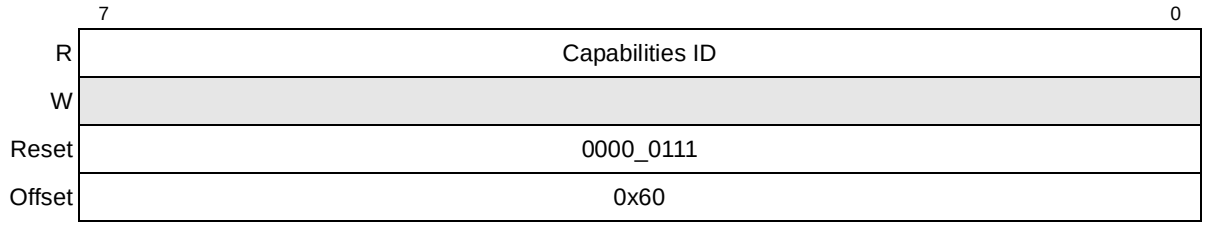


Figure 16-50. PCI-X Next Capabilities ID Register

Table 16-48. 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.

16.3.2.22 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 16-49.

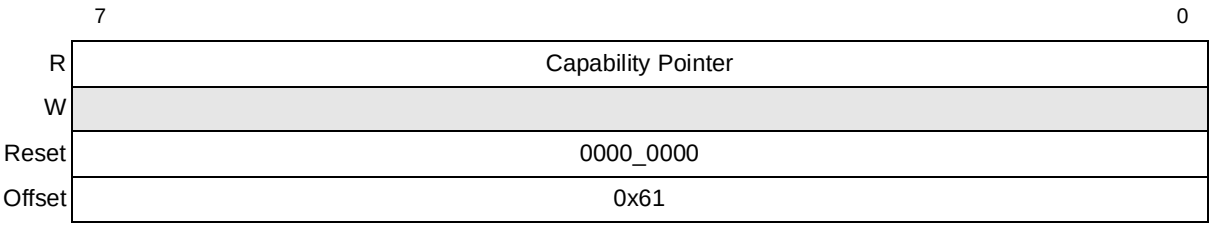


Figure 16-51. PCI-X Capability Pointer Register

Table 16-49. 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.

16.3.2.23 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 16-50 describes the bits of the PCI-X command register.

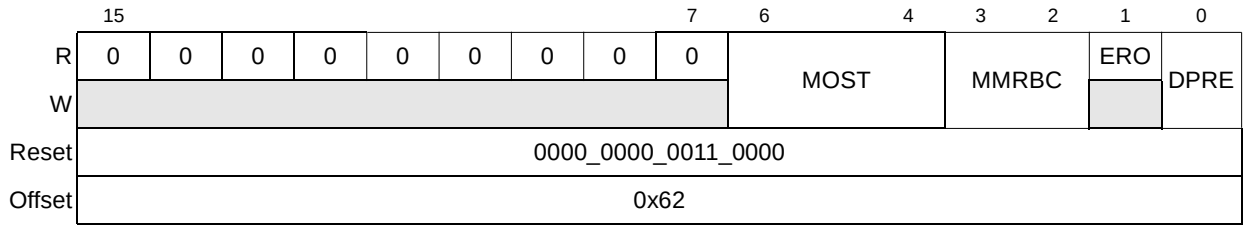


Figure 16-52. PCI-X Command Register

Table 16-50. 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)

16.3.2.24 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 16-51. Only 4-byte accesses to address offset 0x64 are allowed.

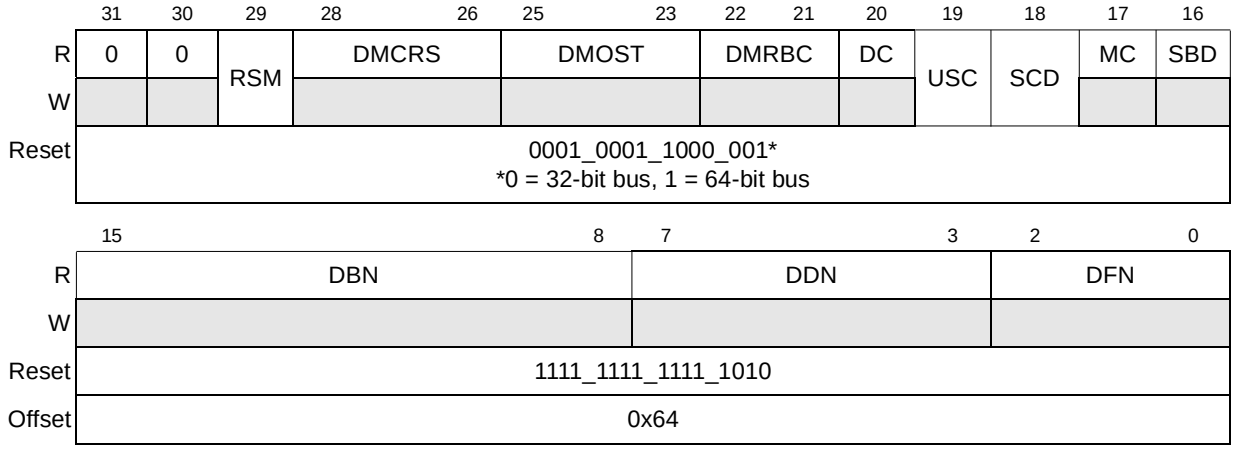


Figure 16-53. PCI-X Status Register

Table 16-51. 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)

16.4 Functional Description

This section describes the functionality of the PCI/X interface.

16.4.1 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 MPC8540 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 MPC8540 governs processor, DMA, and RapidIO accesses to the PCI/X interface. If the on-chip PCI/X arbiter is disabled, the MPC8540 uses the $\overline{\text{PCI\_REQ0}}$ signal as an output to issue its request to the external arbiter and uses the $\overline{\text{PCI\_GNT0}}$ signal as an input to receive its grant from the external arbiter.

The following sections describe the operation of the on-chip PCI/X arbiter that arbitrates between external PCI/X masters and the internal PCI/X bus master of the MPC8540.

16.4.1.1 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 MPC8540, 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 MPC8540 positioned before device 0.

Conceptually, the lowest-priority device is the master that is currently using the bus, and the highest-priority device is the device that follows the current master in numerical order and group priority. This is considered to be a fair algorithm, since a single device cannot prevent other devices from having access to the bus; it automatically becomes the lowest-priority device as soon as it begins to use the bus. If a master is not requesting the bus, then its transaction slot is given to the next requesting device within its priority group.

A grant is awarded to the highest-priority requesting device as soon as the current master begins a transaction; however, the granted device must wait until the bus is relinquished by the current master before initiating a transaction.

The grant given to a particular device may be removed and awarded to another higher-priority device, whenever the higher-priority device asserts its request. If the bus is idle when a device requests the bus, then the arbiter withholds the grant for 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 16-54, 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 16-54, the grant sequence (with all devices, except device 4 requesting the bus and device 3 being the current master) is 0, 2, MPC8540, 0, 2, 1, 0, 2, 3, …, and repeating. If device 2 is not requesting the bus, the grant sequence is 0, MPC8540, 0, 1, 0, 3, …, and repeating. If device 2 requests the bus when device 0 is conducting a transaction and the MPC8540 has the next grant, the MPC8540 has its grant removed and device 2 is awarded the grant since device 2 is higher priority than the MPC8540 when device 0 has the bus.

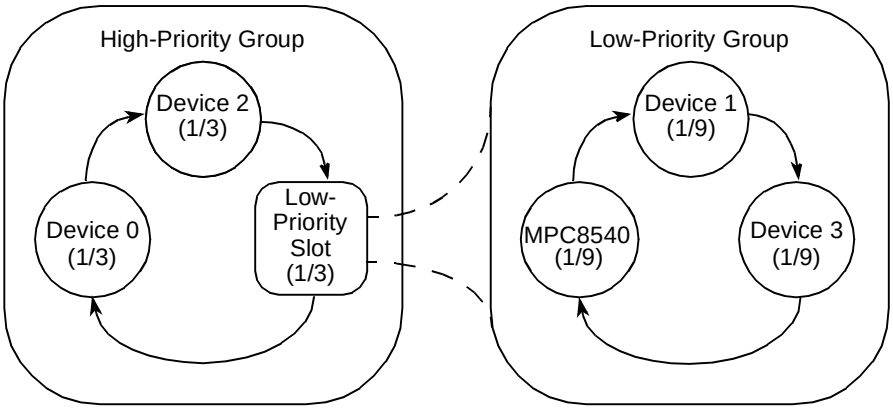


Figure 16-54. PCI/X Arbitration Example

16.4.1.2 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 MPC8540.

16.4.1.3 Broken Master Lock-Out (PCI only)

The PCI bus arbiter on the MPC8540 has a feature that allows it to lock out any masters that are broken or ill-behaved. The broken master feature is controlled by programming bit 12 of the PCI bus arbitration control register (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.

16.4.1.4 Power-Saving Modes and the PCI Arbiter

In the sleep power-saving mode, the clock signal driving SYSCLK can be disabled. If the clock is disabled, 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 MPC8540 when its clocks are stopped, the MPC8540 sustains the PCI_AD[31:0], PCI_C/$\overline{\text{BE}}$[3:0], and parity signals in their prior states. In this situation, the only way for another agent to use the PCI bus is by waking the MPC8540. In the doze or nap power-saving modes, the arbiter continues to operate allowing other PCI devices to run transactions.

16.4.2 PCI Bus Protocol

This section provides a general description of the PCI bus protocol. Specific PCI bus transactions are described in Section 16.4.2.7, “PCI Bus Transactions.” Refer to Figure 16-55, Figure 16-56, Figure 16-57, and Figure 16-58 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 *MPC8540 Integrated Processor Hardware Specifications* for specific setup and hold times.

16.4.2.1 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.

16.4.2.2 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 16-52 describes the PCI bus commands implemented by the MPC8540.

Table 16-52. PCI Bus Commands

PCI_C/BE[3:0]	PCI Bus Command	MPC8540 Supports as an Initiator	MPC8540 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 16.4.2.12.1, "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 16.4.2.12.2, "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 MPC8540 (the target) fetches data from the requested address to the end of the cache line (32 bytes) from local memory, even though all of the data may not be requested by (or sent to) the initiator.
0111	Memory-write	Yes	Yes	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 16.4.2.11, "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 MPC8540 does not respond to these commands.

16.4.2.3 Addressing

PCI defines three physical address spaces—PCI memory space, PCI I/O space, and PCI configuration space. Access to the PCI memory and I/O space is straightforward, although one must take into account the local memory access window and address translation being used. The address translation registers are described in Section 16.3.1, "PCI/X Memory Mapped Registers." Access to the PCI configuration space is described in Section 16.4.2.11, "Configuration Cycles."

Address decoding on the PCI bus is performed by every device for every PCI transaction. Each agent is responsible for decoding its own address. PCI supports two types of address decoding—positive decoding and subtractive decoding. For positive decoding, each device 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 16.4.2.4, "Device Selection," for information about claiming transactions.

The information contained in the two low-order address bits (PCI_AD[1:0]) varies by the address space (memory, I/O, or configuration). Regardless of the encoding scheme, the two low-order address bits are always included in parity calculations.

16.4.2.3.1 Memory Space Addressing

For memory accesses, PCI defines two types of burst ordering controlled by the two low-order bits of the address—linear incrementing (PCI_AD[1:0] = 0b00) and cache wrap mode (PCI_AD[1:0] = 0b10), as shown in Table 16-53. The other two PCI_AD[1:0] possibilities (0b01 and 0b11) are reserved. As an initiator, the MPC8540 always encodes PCI_AD[1:0] = 00 for PCI memory space accesses. As a target, the MPC8540 executes a target disconnect after the first data phase completes if PCI_AD[1:0] = 01 or PCI_AD[1:0] = 0b11 during the address phase of a local memory access. See Section 16.4.2.8.2, "Target-Initiated Termination," for more information on target disconnect conditions.

Table 16-53. Supported Combinations of PCI_AD[1:0]

PCI_AD[1:0]		MPC8540 as Target		MPC8540 as Initiator	
		Read	Write	Read	Write
00	Linear	√	√	√	√
01	Reserved	TD	TD	—	—
10	Cache Wrap	√	TD	—	—
11	Reserved	TD	TD	—	—

For linear incrementing mode, the memory address is encoded/decoded using PCI_AD[63:2]. Thereafter, the address is incremented by 4 bytes after each data phase completes until the transaction is terminated or completed (a 4-byte data width per data phase is implied). Note that the two low-order bits on the address bus are included in all parity calculations.

For cache wrap mode (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 MPC8540 does not support cache-wrap write operations and executes a target disconnect after the data phase for the end of the cache line completes for writes with PCI_AD[1:0] = 0b10. That is, the MPC8540 does not wrap back to the beginning of the cache line. Note that the two low-order bits on the address bus are included in all parity calculations.

16.4.2.3.2 I/O Space Addressing

For PCI I/O accesses, 32 address signals (PCI_AD[31:0]) are used to provide a byte address. After a target has claimed an I/O access, it must determine if it can complete the entire access as indicated by the byte enable signals. If all the selected bytes are not in the address range of the target, the entire access cannot complete. In this case, the target does not transfer any data and terminates the transaction with a target-abort error. See Section 16.4.2.8.2, “Target-Initiated Termination,” for more information.

16.4.2.3.3 Configuration Space Addressing

PCI supports two types of configuration accesses that use different formats for the 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 16.4.2.11, “Configuration Cycles,” for descriptions of the two formats.

16.4.2.4 Device Selection

The $\overline{\text{PCI\_DEVSEL}}$ signal is driven by the target of the current transaction. $\overline{\text{PCI\_DEVSEL}}$ indicates to the other devices on the PCI bus that the target has decoded the address and claimed the transaction. $\overline{\text{PCI\_DEVSEL}}$ may be driven one, two, or three clock cycles (fast, medium, or slow device select timing) following the address phase. Device select timing is encoded into the device’s PCI 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 MPC8540 is hardwired for fast device select timing (PCI bus status register [10–9] = 0b00). Therefore, when the MPC8540 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 MPC8540 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 16.4.2.8.1, "Master-Initiated Termination."

16.4.2.5 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 16.4.2.13.1, "PCI Parity," for more information.

If the MPC8540, as a target, detects no byte enables asserted, it completes the current data phase with no permanent change. This implies that on a read transaction, the MPC8540 expects that the data is not changed, and on a write transaction, the data is not stored.

16.4.2.6 Bus Driving and Turnaround

To avoid contention, a turnaround cycle is required on all signals that may be driven by more than one agent. The turnaround cycle occurs at different times for different signals. The 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 16.4.2.13.1, "PCI Parity," for more information.

16.4.2.7 PCI Bus Transactions

This section provides descriptions of the PCI bus transactions. All bus transactions follow the protocol as described in Section 16.4.2, "PCI Bus Protocol." Read and write transactions are

similar for the memory and I/O spaces, so they are described as generic read transactions and generic write transactions.

The timing diagrams in this section show the relationship of significant signals involved in bus transactions. When a signal is drawn as a solid line, it is actively being driven by the current master or target. When a signal is drawn as a dashed line, no agent is actively driving it. High-impedance signals are indicated to have indeterminate values when the dashed line is between the two rails.

The terms ‘edge’ and ‘clock edge’ always refer to the rising edge of the clock; ‘asserted’ and ‘negated’ always refer to the globally visible state of the signal on the clock edge, and not to signal transitions. ‘ ’ represents a turnaround cycle in the timing diagrams.

16.4.2.7.1 PCI Read Transactions

This section describes PCI single-beat read transactions and PCI burst read transactions.

A read transaction starts with the address phase, occurring when an initiator asserts PCI_FRAME. During the address phase, PCI_AD[63:0] contains a valid address and PCI_C/BE[7:0] contains a valid bus command.

The first data phase of a read transaction requires a turnaround cycle. This allows the transition from the initiator driving PCI_AD[63:0] as address signals to the target driving PCI_AD[63:0] as data signals. The turnaround cycle is enforced by the target with the TRDY signal. The target provides valid data at the earliest one cycle after the turnaround cycle. The target must drive the PCI_AD[63:0] signals when PCI_DEVSEL is asserted.

During the data phase, the PCI_C/BE[7:0] signals indicate which byte lanes are involved in the current data phase. A data phase may consist of a data transfer and wait cycles. The PCI_C/BE[7:0] signals remain actively driven for both reads and writes from the first clock of the data phase through the end of the transaction.

A data phase completes when data is transferred, which occurs when both PCI_IRDY and PCI_TRDY are asserted on the same clock edge. When either PCI_IRDY or PCI_TRDY is negated, a wait cycle is inserted and no data is transferred. The initiator indicates the last data phase by negating PCI_FRAME when PCI_IRDY is asserted. The transaction is considered complete when data is transferred in the last data phase.

Figure 16-55 illustrates a PCI single-beat read transaction.

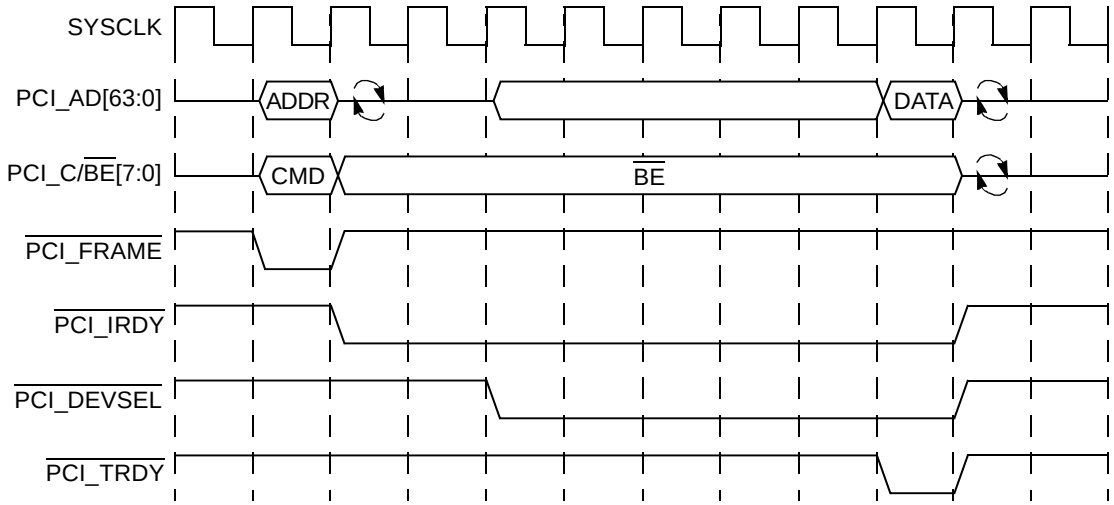


Figure 16-55. PCI Single-Beat Read Transaction

Figure 16-56 illustrates a PCI burst read transaction.

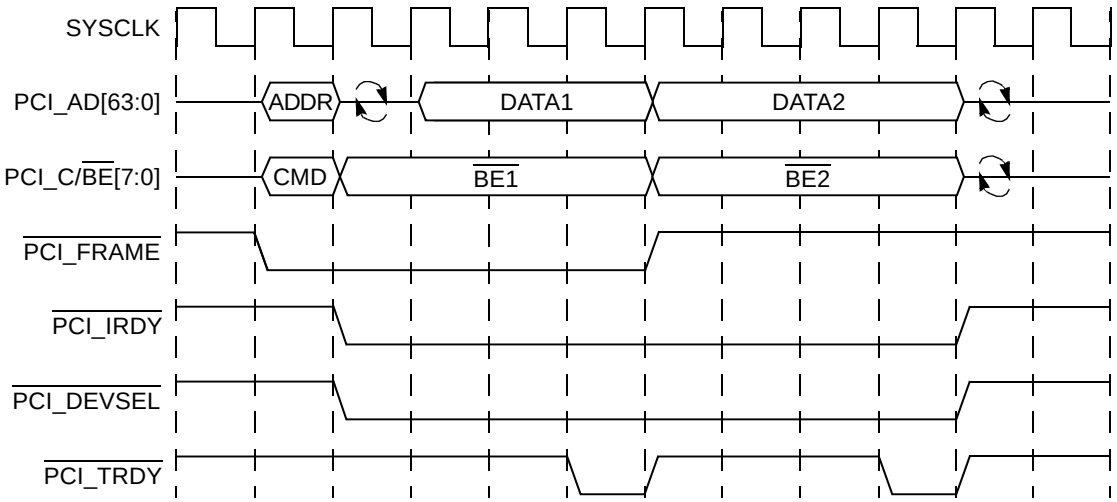


Figure 16-56. PCI Burst Read Transaction

16.4.2.7.2 PCI Write Transactions

This section describes PCI single-beat write transactions, and PCI burst write transactions. A PCI write transaction starts with the address phase, occurring when an initiator asserts PCI_FRAME. A write transaction is similar to a read transaction except no turnaround cycle is needed following the address phase because the initiator provides both address and data. The data phases are the same for both read and write transactions. Although not shown in the figures, the initiator must

drive the PCI_C/$\overline{\text{BE}}$[7:0] signals, even if the initiator is not ready to provide valid data ($\overline{\text{PCI\_IRDY}}$ negated).

Figure 16-57 illustrates a PCI single-beat write transaction.

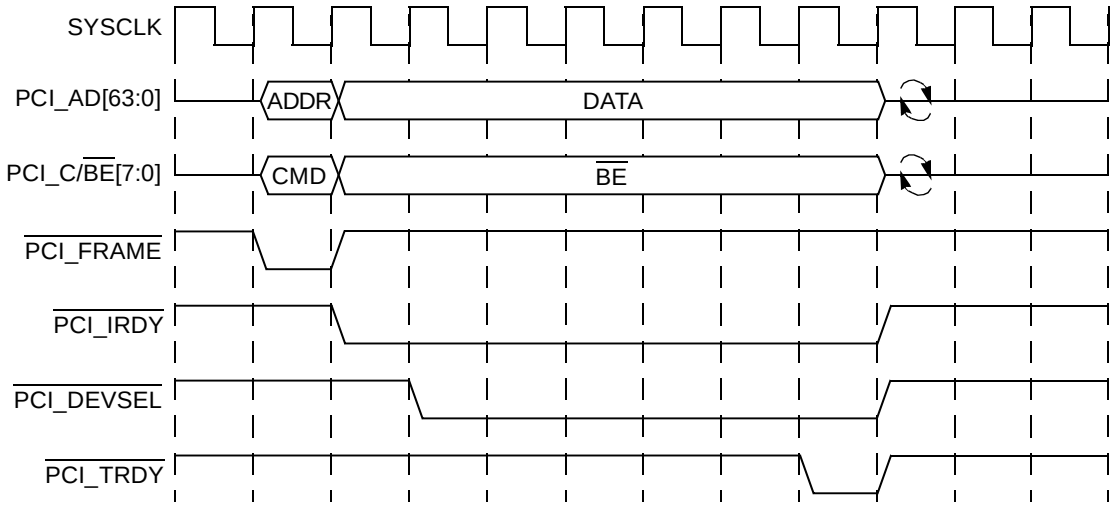


Figure 16-57. PCI Single-Beat Write Transaction

Figure 16-58 illustrates a PCI burst write transaction.

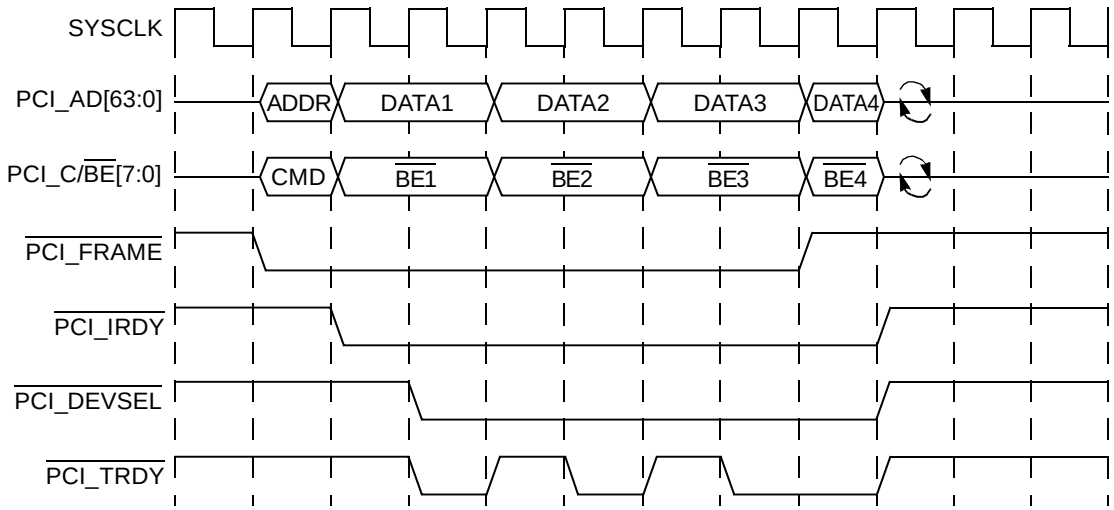


Figure 16-58. PCI Burst Write Transaction

16.4.2.8 Transaction Termination

A PCI transaction may be terminated by either the initiator or the target. The initiator is ultimately responsible for concluding all transactions, regardless of the cause of the termination. All

transactions are concluded when PCI_FRAME and PCI_IRDY are both negated, indicating the bus is idle.

16.4.2.8.1 Master-Initiated Termination

Normally, a master initiates termination by negating 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 MPC8540 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.

16.4.2.8.2 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 MPC8540 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 16.4.2.3.1, “Memory Space Addressing.”

As a target, the MPC8540 responds to a transaction with a retry due to the following:

- The 16-clock latency timer has expired, and the first data phase has not begun.
- There is no more internal buffer space available for an inbound transaction.

Target-abort is indicated by asserting $\overline{\text{PCI\_STOP}}$ and negating $\overline{\text{PCI\_DEVSEL}}$. This indicates that the target requires termination of the transaction and does not want the transaction retried. If a transaction is terminated by target-abort, the received target-abort bit (bit 12) of the initiator’s 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 MPC8540 aborts the transaction internally, but continues the transaction on the PCI bus.

Figure 16-59 shows several target-initiated terminations.

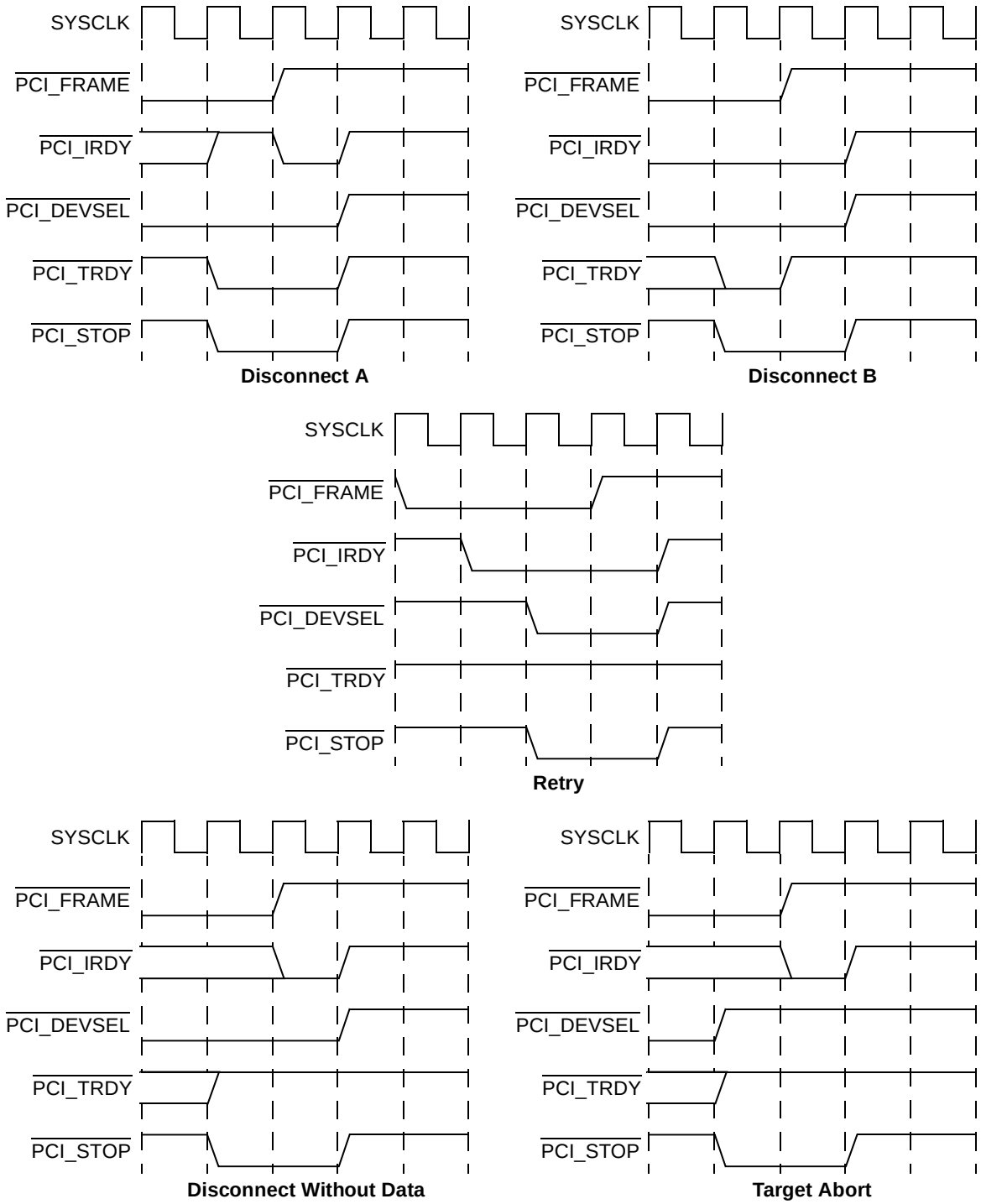


Figure 16-59. PCI Target-Initiated Terminations

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.

16.4.2.9 Fast Back-to-Back Transactions

The PCI bus allows fast back-to-back transactions by the same master. During a fast back-to-back transaction, the initiator starts the next transaction immediately without an idle state. The last data phase completes when 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 MPC8540 does not perform any fast back-to-back transactions. As a target, the MPC8540 supports both types of fast back-to-back transactions.

During fast back-to-back transactions, the MPC8540 monitors the bus states to determine if it is the target of a transaction. If the previous transaction was not directed to the MPC8540 and the current transaction is directed at the MPC8540, it delays the assertion of 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.

16.4.2.10 Dual Address Cycles

The MPC8540 supports dual address cycle (DAC) commands (64-bit addressing on PCI bus) as both an initiator and a target. DACs are different from single address cycles (SACs) in that the address phase takes two PCI beats instead of one PCI beat to transfer (64-bit vs. 32-bit addressing). Only PCI memory commands can use DAC cycles; I/O, configuration, interrupt acknowledge, and special cycle command cannot use DAC cycles. The MPC8540 block supports single-beat and burst DAC transactions.

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 16.3.1.2, “PCI/X ATMU Outbound Registers,” for more information.

Figure 16-60 shows the timing sequence of the PCI signals for single-beat DAC reads.

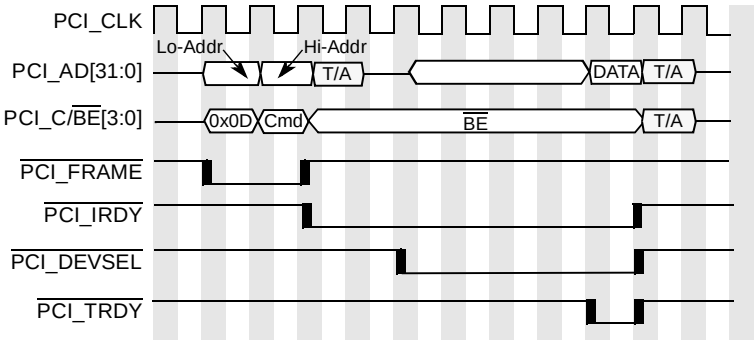


Figure 16-60. DAC Single-Beat Read Example

The timing for a DAC burst read is shown in Figure 16-61.

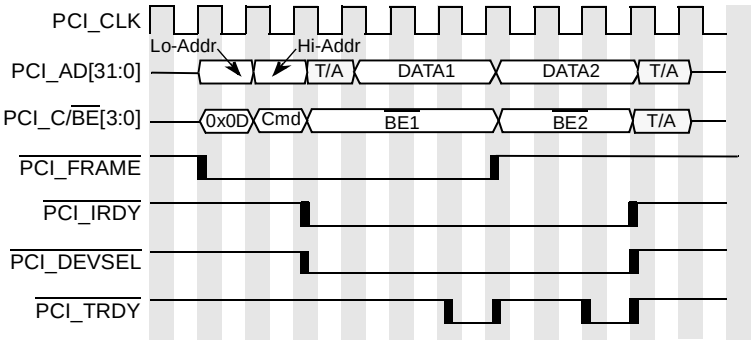


Figure 16-61. DAC Burst Read Example

Figure 16-62 and Figure 16-63 show single-beat DAC writes and burst DAC writes.

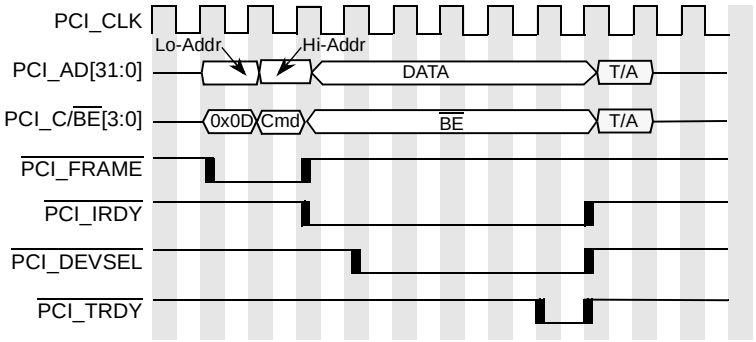


Figure 16-62. DAC Single-Beat Write Example

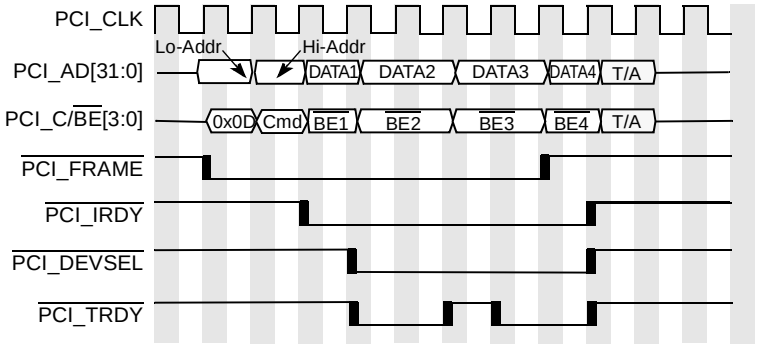


Figure 16-63. DAC Burst Write Example

Figure 16-64 shows the timing sequence of the PCI signals for 64-bit DAC reads.

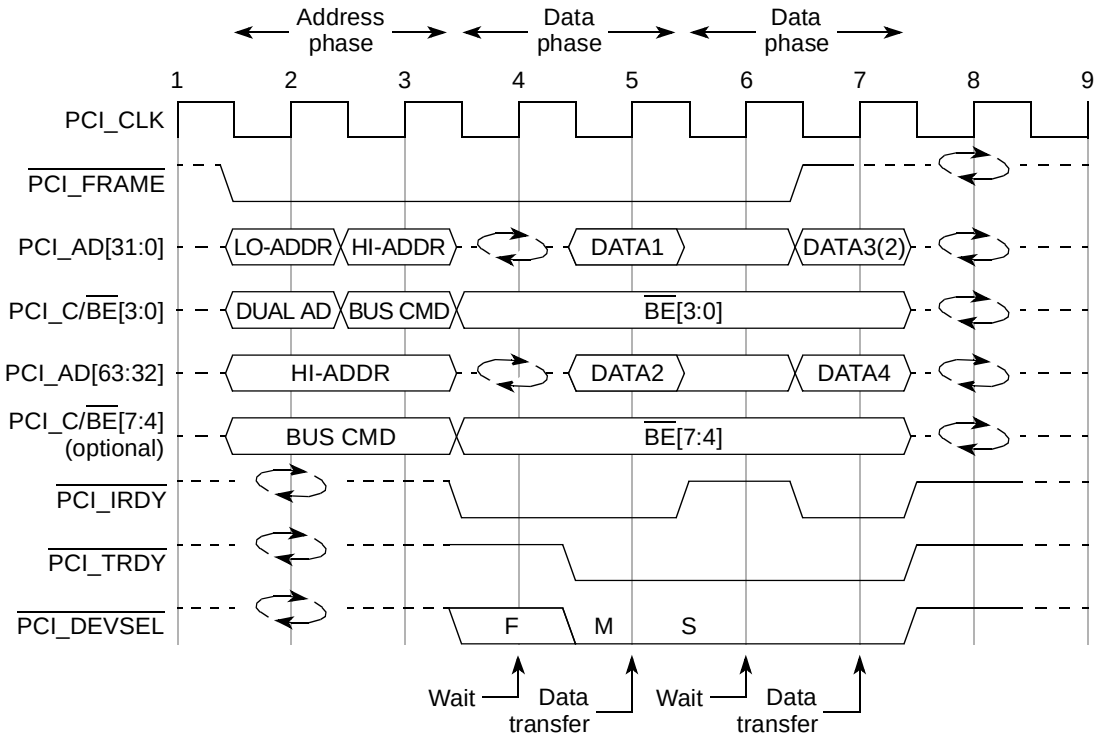


Figure 16-64. 64-Bit Dual Address Read Cycle

16.4.2.11 Configuration Cycles

This section describes PCI configuration cycles used for configuring standard PCI devices. The PCI configuration space of any device is intended for configuration, initialization, and catastrophic error-handling functions only. Access to the PCI configuration space should be limited to initialization and error-handling software.

16.4.2.11.1 PCI Configuration Space Header

Figure 16-65 shows the predefined header for all PCI devices.

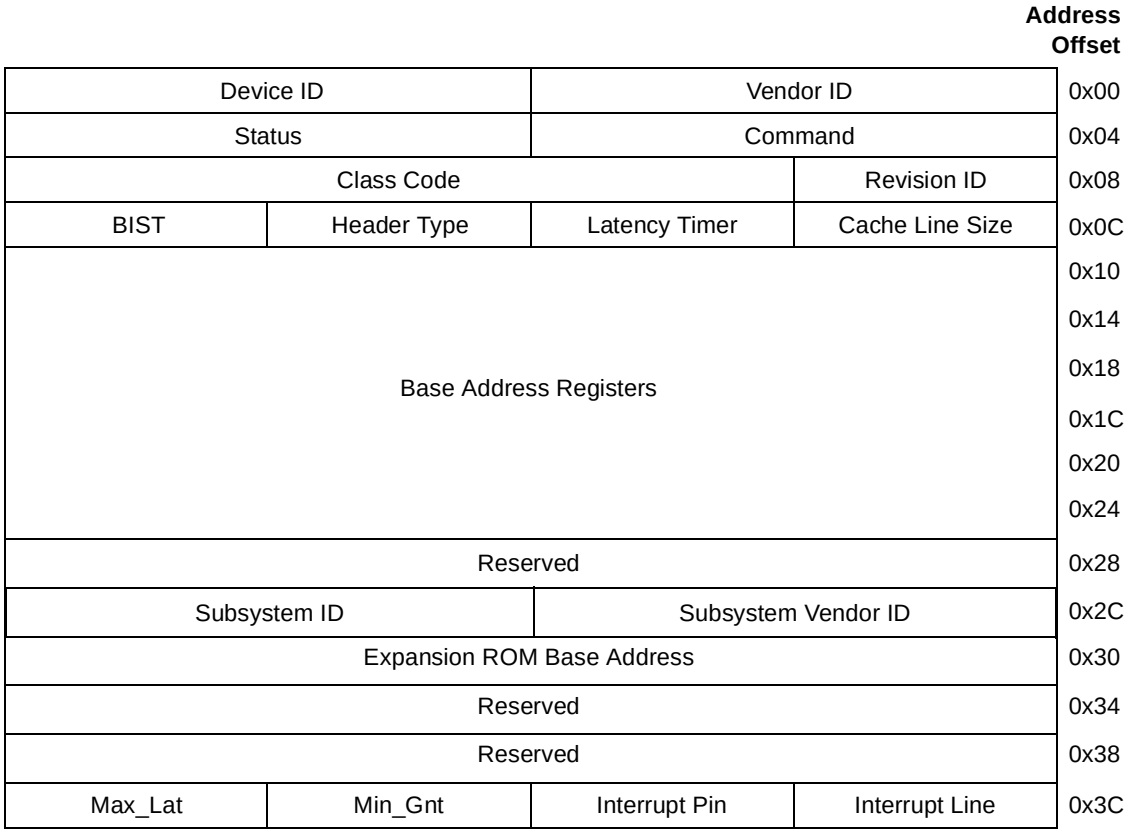


Figure 16-65. Standard PCI Configuration Header

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 16-65. The rest of the 256-byte configuration space is device-specific. The PCI header specific to the MPC8540 is described in Section 16.3.2, "PCI/X Configuration Header."

Table 16-54 summarizes the configuration header registers. Detailed descriptions of these registers are provided in the *PCI Local Bus Specification*, Rev. 2.2.

Table 16-54. 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 16-65 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 MPC8540)
0x2E	Subsystem ID	Identifies the subsystem ID (read-only for MPC8540)
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

16.4.2.11.2 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 MPC8540's PCI bus command register must be set before an outbound configuration access is attempted. Device 0 on PCI/X bus 0 is the MPC8540 itself; thus, device 0, bus 0 is used to access the internal PCI/X configuration header.

When the MPC8540 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 MPC8540 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 MPC8540 performs a type 0 configuration cycle translation. If the bus number indicates a remote PCI bus (that is, nonlocal), the MPC8540 performs a type 1 configuration cycle translation. 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 16.3.1.1.1, "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 MPC8540 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)
```

16.4.2.11.3 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 MPC8540 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 MPC8540 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.

16.4.2.11.4 PCI Type 0 Configuration Translation

Figure 16-66 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.

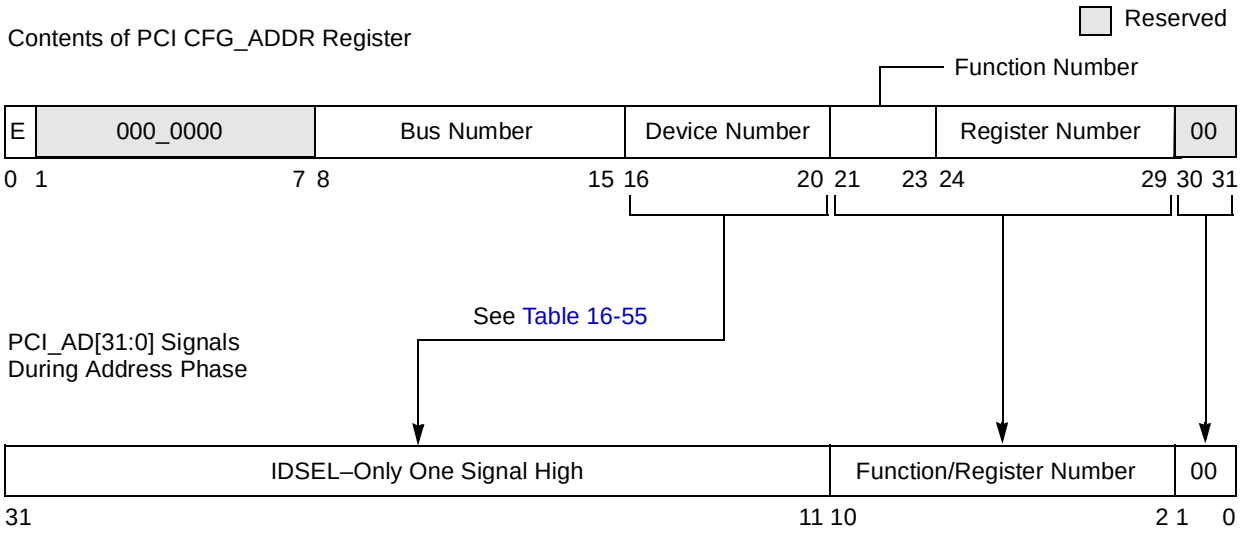


Figure 16-66. PCI Type 0 Configuration Translation

For PCI type 0 configuration cycles, the MPC8540 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 MPC8540 translates the device number to AD*n* as shown in Table 16-55.

Table 16-55. 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 MPC8540 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 $\overline{\text{PCI\_FRAME}}$.

16.4.2.11.5 Type 1 Configuration Translation

For type 1 translations, the MPC8540 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 MPC8540 automatically translates PCI_AD[1:0] into 0b01 during the address phase to indicate a type 1 configuration cycle.

16.4.2.12 Other Bus Transactions

There are two other PCI transactions that the MPC8540 supports—interrupt acknowledge and special cycles. As an initiator, the MPC8540 may initiate both interrupt acknowledge and special-cycle transactions; however, as a target, the MPC8540 ignores interrupt-acknowledge and special-cycle transactions. Both transactions make use of the PCI CFG_ADDR and PCI CFG_DATA registers described in Section 16.4.2.11.3, "PCI Configuration in Agent and Agent Lock Modes."

16.4.2.12.1 Interrupt-Acknowledge Transactions

The PCI bus supports an interrupt-acknowledge transaction. The interrupt-acknowledge command is a read operation implicitly addressed to the system interrupt controller. Note that the PCI interrupt-acknowledge command does not address the MPC8540 PIC processor interrupt-acknowledge register and does not return the interrupt vector address from the PIC unit. See Chapter 10, "Programmable Interrupt Controller," for more information about the PIC unit.

When the MPC8540 detects a read to 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 MPC8540 performs an interrupt-acknowledge transaction. If the bus number indicates a nonlocal PCI bus, the MPC8540 performs a type 1 configuration cycle translation, similar to any other configuration cycle for which the bus number does not match.

The address phase contains no valid information other than the interrupt-acknowledge command (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 MPC8540 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 MPC8540 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.

16.4.2.12.2 Special-Cycle Transactions

The special-cycle command provides a mechanism to broadcast select messages to all devices on the PCI bus. The special-cycle command contains no explicit destination address but is broadcast to all PCI agents.

When the MPC8540 detects a write to 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 MPC8540 performs a special-cycle transaction on the local PCI bus. If the bus number indicates a nonlocal PCI bus, the MPC8540 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 16-56.

Table 16-56. Special-Cycle Message Encodings

PCI_AD[15:0]	Message
0x0000	SHUTDOWN
0x0001	HALT
0x0002	x86 architecture-specific
0x0003–0xFFFF	—

Note that the MPC8540 does not automatically issue a special-cycle message when it enters any of its power-saving modes. It is the responsibility of software to issue the appropriate special-cycle message, if needed.

Each receiving agent must determine whether the special-cycle message is applicable to itself. 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.

16.4.2.13 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.

16.4.2.13.1 PCI Parity

Generating parity is not optional; it must be performed by all PCI-compliant devices. All PCI transactions, regardless of type, calculate even parity; that is, the number of ones on the 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 16-67.

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.

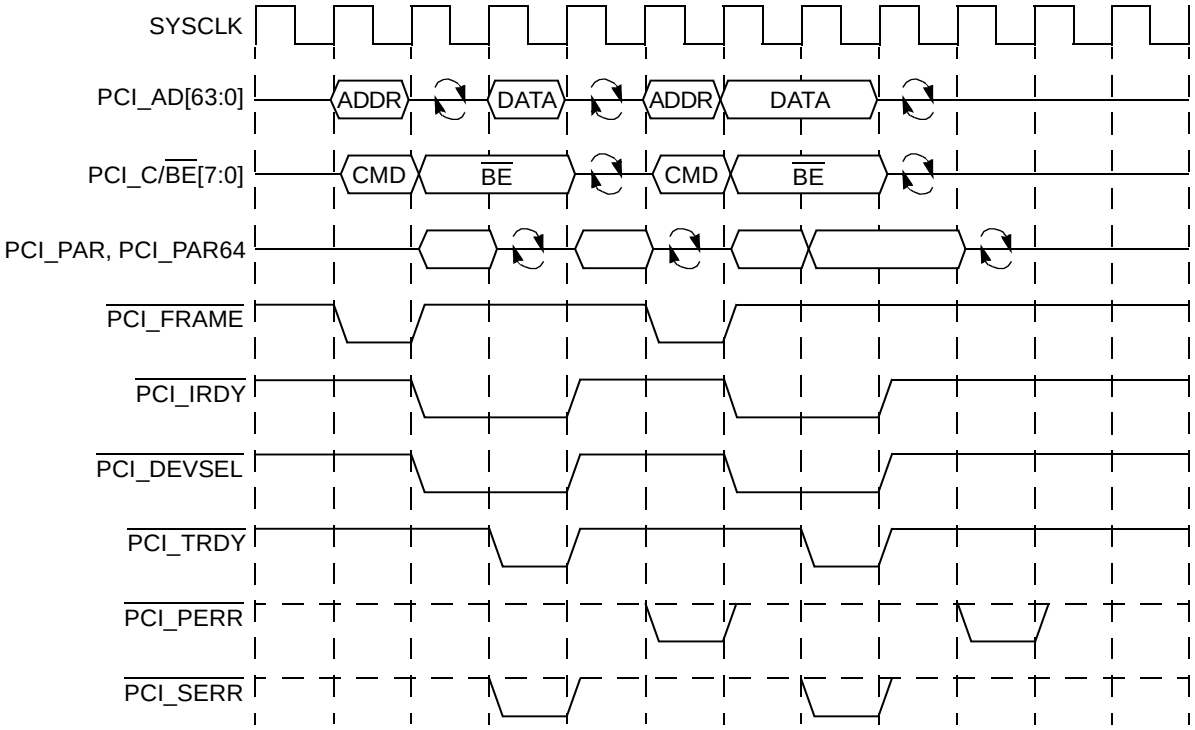


Figure 16-67. PCI Parity Operation

16.4.2.13.2 Error Reporting

PCI provides for the detection and signaling of both parity and other system errors. Two signals are used to report these errors—PCI_PERR and PCI_SERR. The PCI_PERR signal is used exclusively to report data parity errors on all transactions except special cycles. The PCI_SERR signal is used for other error signaling including address parity errors and data parity errors on special-cycle transactions; it may also be used to signal other system errors.

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 16-57 shows the actions taken for each kind of error.

Table 16-57. 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

Table 16-57. PCI Mode Error Actions (continued)

PCI Error Type	Error Detect Register Bit	PCI Bus Status Register Bit	Comment
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

16.4.3 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.

16.4.3.1 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.

16.4.3.2 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 16-52 describes the PCI-X bus commands implemented by the MPC8540.

Table 16-58. PCI-X Command Encodings

PCI_C/$\overline{\text{BE}}$[3:0]	PCI -X Bus Command	Length/ Byte-Enable Usage	MPC8540 Supports as Initiator	MPC8540 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 16.4.2.12.1, "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 16.4.2.12.2, "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 16.4.3.7, "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 16.4.3.7, "PCI-X Configuration Transactions," for more detail on PCI-X configuration cycles.

Table 16-58. PCI-X Command Encodings (continued)

PCI_C/BE[3:0]	PCI -X Bus Command	Length/ Byte-Enable Usage	MPC8540 Supports as Initiator	MPC8540 Supports as Target	Definition
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.
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 MPC8540 does not respond to these commands.

16.4.3.3 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 16-68 shows PCI_AD[31:0] attributes for burst and DWORD transactions.

The attributes for split completion transactions are described in Section 16.4.3.6, “PCI-X Split Transactions,” and the attributes for PCI-X configuration transactions are described in Section 16.4.3.7, “PCI-X Configuration Transactions.”

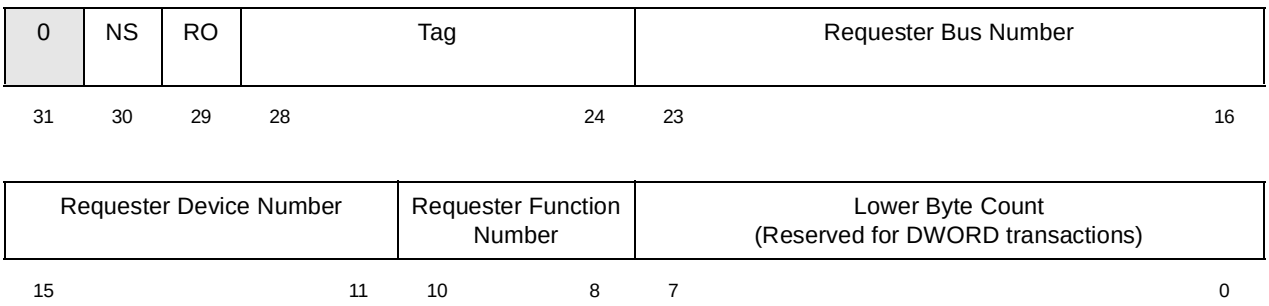


Figure 16-68. AD[31:0] During PCI-X Burst/DWORD Attribute Phase

Table 16-59 describes the fields of the attribute for burst and DWORD transactions.

Table 16-59. 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.
29	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.

16.4.3.4 PCI-X Transactions

Figure 16-69 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 16-58.

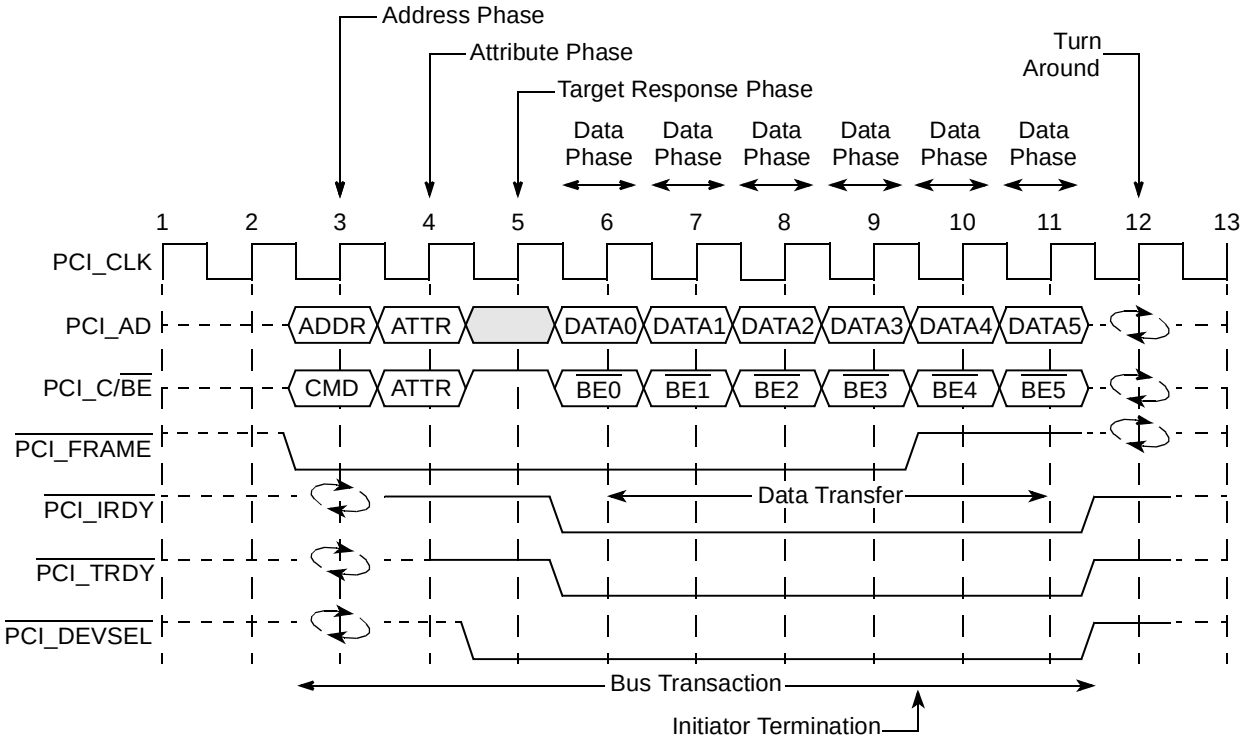


Figure 16-69. Typical PCI-X Write Transaction

Figure 16-70 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.

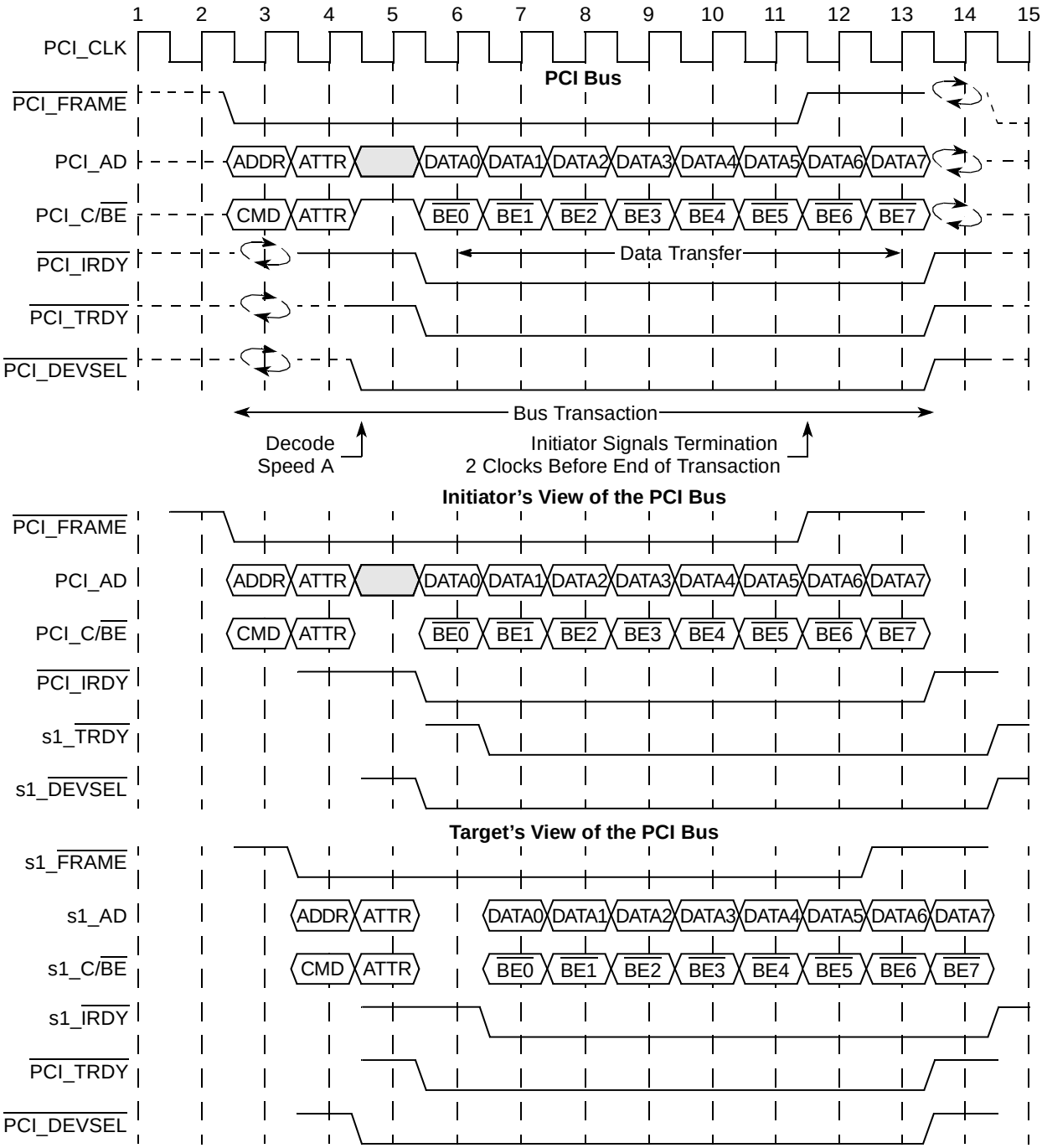


Figure 16-70. Burst Memory Write Transaction

Figure 16-71 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.

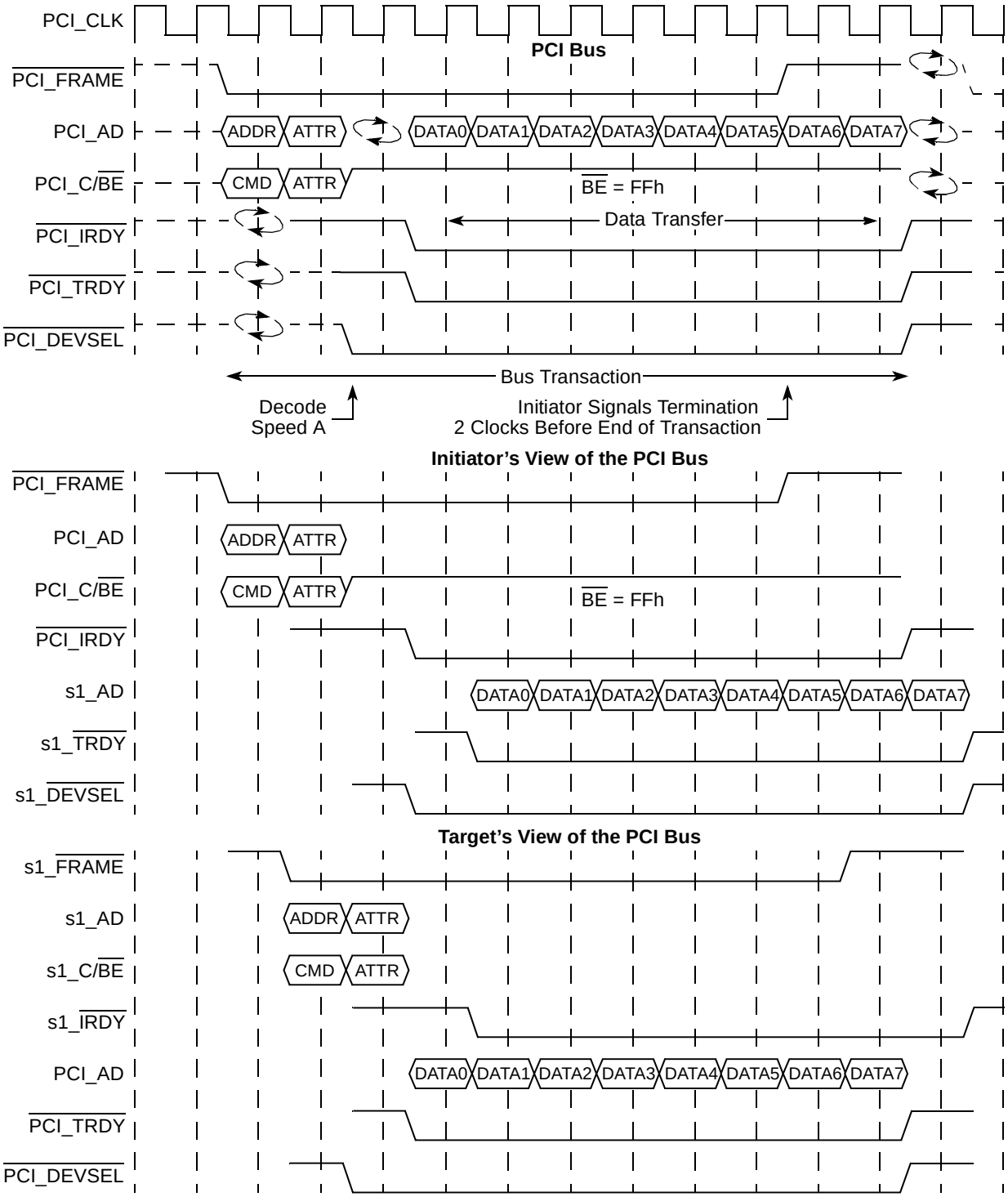


Figure 16-71. Memory Read Block Transaction

Figure 16-72 shows a DWORD write transaction such as an I/O write. Note that there are no target initial wait states for this example.

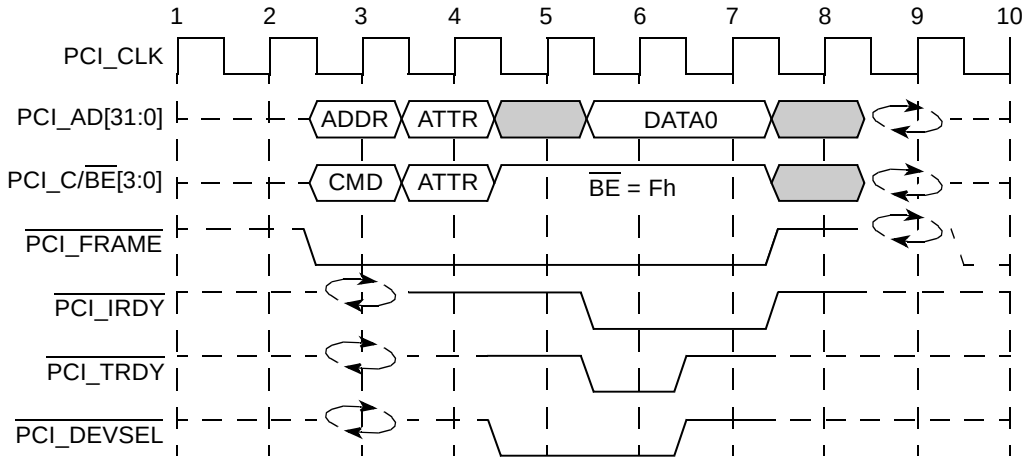


Figure 16-72. DWORD (4-Byte) Write Transaction

Figure 16-73 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.

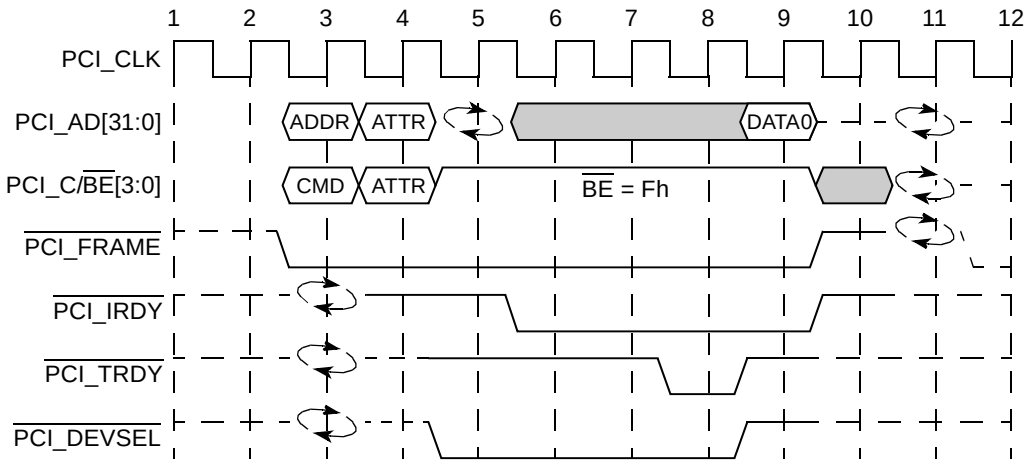


Figure 16-73. DWORD (4-Byte) Read Transaction

16.4.3.5 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 MPC8540 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 16.4.2.8.2, "Target-Initiated Termination," the MPC8540 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.

16.4.3.6 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.

16.4.3.6.1 Split Response

Figure 16-74 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.

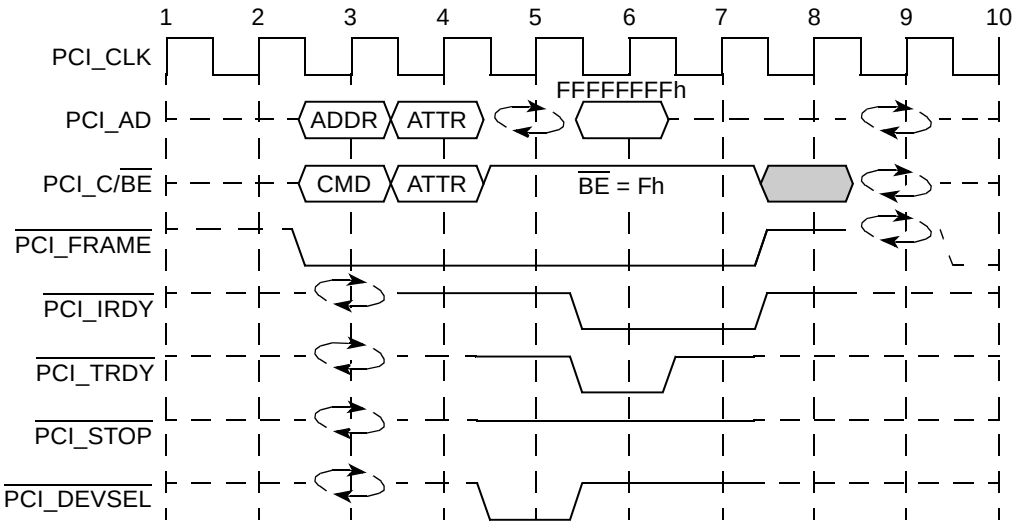


Figure 16-74. Split Response to a Read Transaction

Figure 16-74 shows a split response to a write transaction.

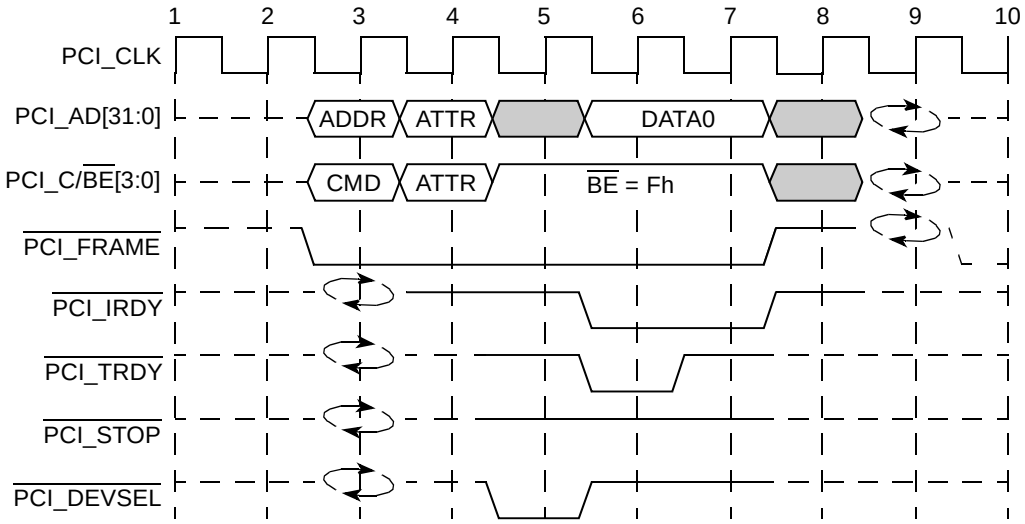


Figure 16-75. Split Response to a DWORD (4-Byte) Write Transaction

16.4.3.6.2 Completion Address

Figure 16-76 shows how AD[31:0] are driven during the address phase for split completion transactions.

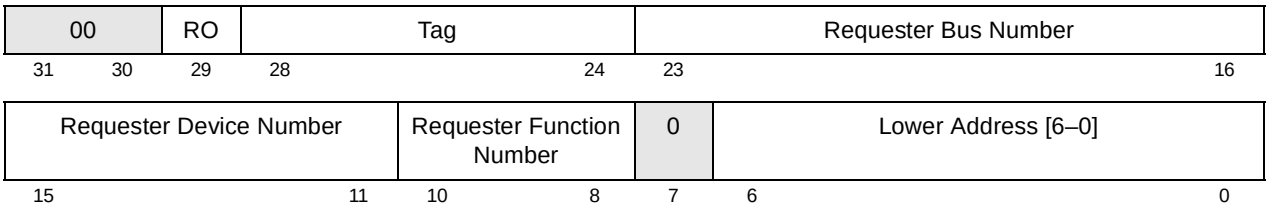


Figure 16-76. Split Completion Transaction Address AD[31:0]

Table 16-60 describes the fields of the attribute for Burst and DWORD transactions.

Table 16-60. Split Completion Transaction Address

AD	Name	Description
31:30	—	Reserved. Must be 00.
29	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.

16.4.3.6.3 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 16-77.

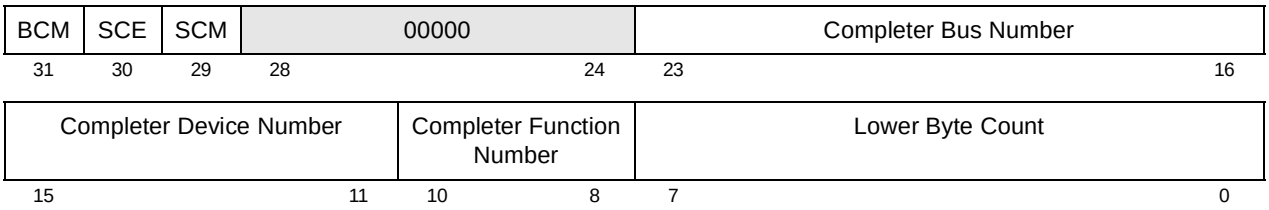


Figure 16-77. AD[31:0] During PCI-X Split Completion Attribute Phase

Table 16-61 describes the fields of the completer attribute for PCI-X split completion transactions.

Table 16-61. 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.

16.4.3.6.4 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 16-78.

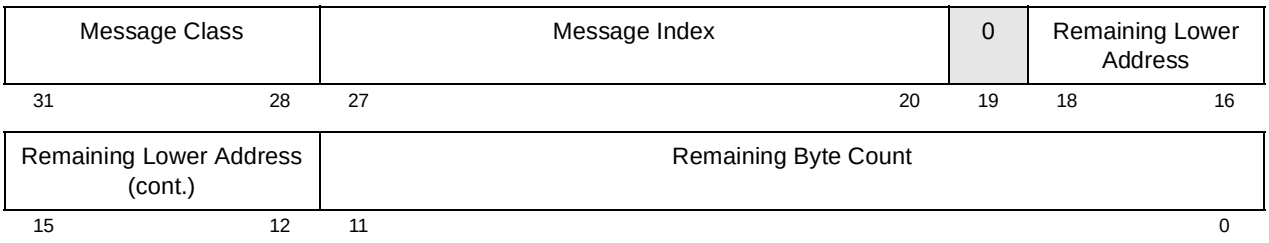


Figure 16-78. PCI-X Split Completion Message Format

Table 16-62 describes the fields of the PCI-X split completion message.

Table 16-62. 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 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 MPC8540 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.

16.4.3.7 PCI-X Configuration Transactions

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:

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 MPC8540's PCI bus command register must be set before an outbound configuration access is attempted. Figure 16-79 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.

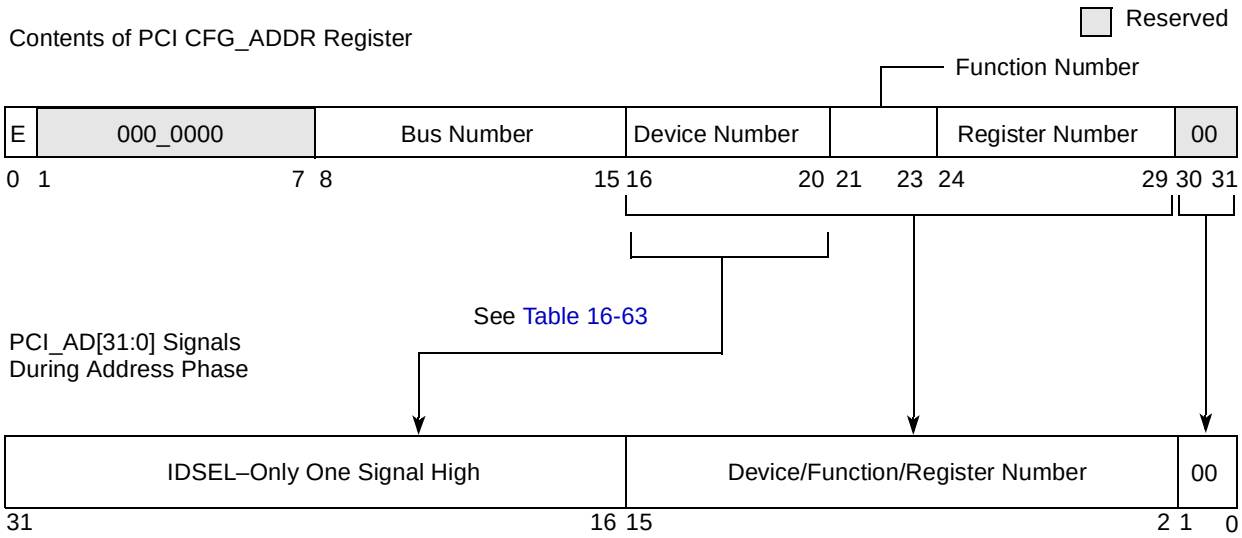


Figure 16-79. PCI-X Type 0 Configuration Translation

Compare the PCI-X translation in Figure 16-79 with the PCI configuration translation shown in Figure 16-66. 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 MPC8540 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 MPC8540 translates the device number to AD*n* as shown in Table 16-63.

Table 16-63. 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 MPC8540 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 16-80.

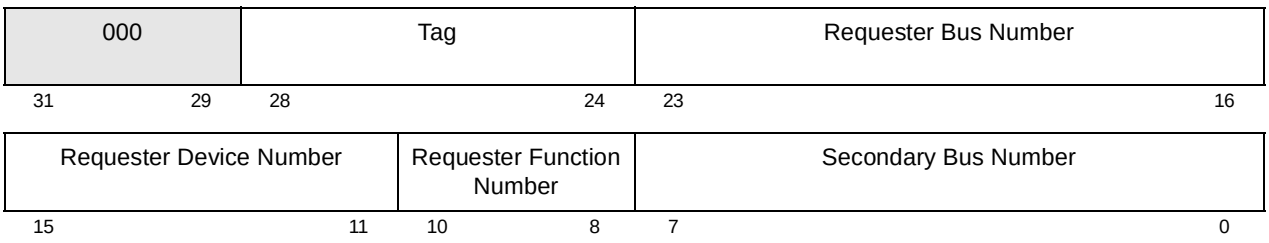


Figure 16-80. AD[31:0] During PCI-X Configuration Attribute Phase

Table 16-64 describes the fields of the attribute for PCI-X configuration transactions.

Table 16-64. 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

16.4.3.8 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.

16.4.3.8.1 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 16-65 shows actions taken for each kind of error.

Table 16-65. 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 16-65. 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.

16.5 Initialization/Application Information

This section describes some tips for use of the PCI/X controller.

16.5.1 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 16-66.

Table 16-66. 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 16-67.

Table 16-67. 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

16.5.1.1 Host Mode

When the device powers up in host mode, all inbound configuration accesses are ignored (and thus master aborted). The ACL bit is a don't care. The 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.

16.5.1.2 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.

16.5.1.3 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.

16.5.2 Nonposted Writes in PCI-X

The PCI-X block may pipeline I/O write and configuration write accesses to a target that signals a split response. 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.

16.5.3 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 MPC8540 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 MPC8540 when writing software to meet these alignment restrictions.

16.5.4 PCI-X Inbound Reads that Cross Window Boundaries

For Rev 1 of the MPC8540, 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 MPC8540 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.

16.5.5 Bridging Between RapidIO and PCI/X

The MPC8540 does not support bridging from RapidIO according to the RapidIO Interoperability Specification. If an external RapidIO device tries to target the MPC8540 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 MPC8540 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 MPC8540s communicating with each other via PCI or PCI-X may deadlock when two RapidIO devices (one on each MPC8540) are simultaneously performing priority-1 writes to targets on the remote MPC8540 through the PCI/X link between the devices. To support such a topology, RapidIO write transactions that bridge to the other MPC8540 must be priority 0.

Chapter 17
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.

17.1 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.

17.1.1 Overview

A block diagram of The RapidIO controller is shown in Figure 17-1 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.

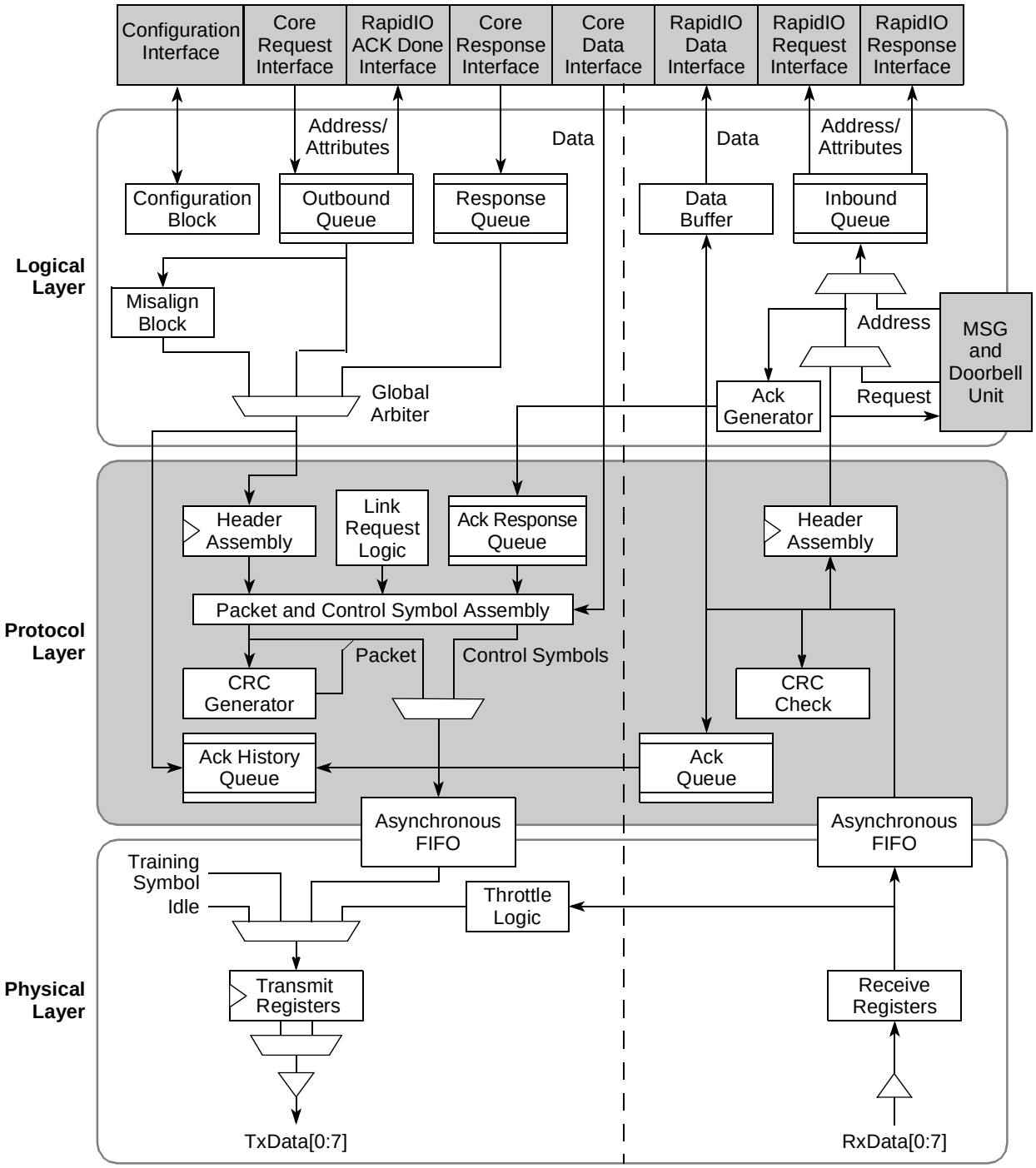


Figure 17-1. RapidIO High Level Block Diagram

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 17.2, “External Signal Descriptions,” describes the interface signal groups, their meanings, and timing relationships.
- Section 17.3, “Memory Map/Register Definition,” describes the configuration, runtime, and debug registers that constitute the programming model.
- Section 17.4.1, “RapidIO Transaction, Packet, and Control Symbol Summary,” describes the functional organization and operation of the design.
- Section 17.5, “RapidIO Functionality,” provides general information about implementing RapidIO features.

17.1.2 RapidIO Terminology

Table 17-1 describes terminology used throughout this chapter.

Table 17-1. 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

17.1.3 RapidIO Interface Features

This section describes the RapidIO interface features.

17.1.3.1 Supported Features

RapidIO supports the following features of the *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

17.1.3.2 Features Not Implemented

The following portions of the *Parallel RapidIO Interconnect 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

17.1.4 RapidIO Modes of Operation

This section describes the RapidIO modes of operation.

17.1.4.1 Transmit Clock-Select Mode

The RapidIO transmit clock can be chosen from either an internal source or from the RapidIO receive clock 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.

17.1.4.2 CRC Checking Modes

RapidIO offers the modes for packet CRC checking shown in Table 17-2. 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 17-2. CRC Checking Mode

PCR[CCE]	Mode
0	CRC checking disabled
1	CRC enabled with hardware-based recovery

17.1.4.3 Error Checking Disable Mode

Error checking can be enabled or disabled by setting a bit in the port control CSR. This mode disables all RapidIO transmission error checking. Device behavior is undefined if error checking is disabled and an error condition occurs.

17.1.4.4 Output Port Enable Mode

The output port can be disabled by setting a bit in the port control CSR. In this mode, RapidIO does not issue any packets of its own, except in response to maintenance packets.

17.1.4.5 Output Port Driver Disable Mode

This mode, controlled by PCCSR[OPD], disables RapidIO input receivers and output drivers.

17.1.4.6 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 17-3 describes the behavior with respect to the accept all mode.

Table 17-3. 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

17.1.4.7 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 17.3.2.3.2, “Port Notification/Fatal Error Detect Disable Register (PNFEDiR).”

17.1.4.8 Link Response Time-Out Disable Mode

The RapidIO link time-out counters can be disabled by setting a bit in the RapidIO port notification/fatal error disable register. For more information, see Section 17.3.2.3.2, “Port Notification/Fatal Error Detect Disable Register (PNFEDiR).”

17.2 External Signal Descriptions

This section describes the signal names of the RapidIO block, their meanings, and the timing relationships that exist among the signals.

17.2.1 Signals Overview

The RapidIO block implements the functionality described in the *Parallel RapidIO Interconnect Specification* for an 8-bit parallel interface with separate transmit and receive ports as shown in Figure 17-2.

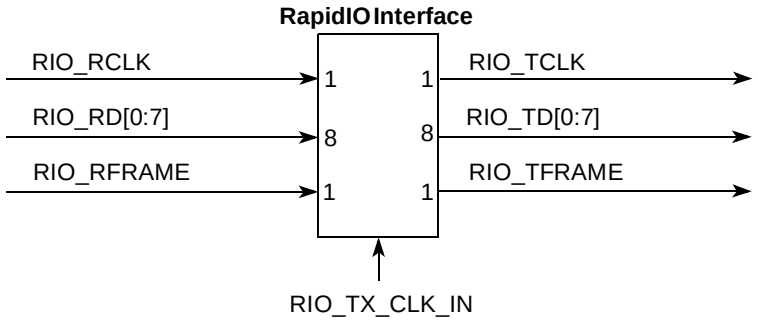


Figure 17-2. RapidIO Interface Signals Model

The RapidIO signals are implemented as differential pairs of signals. Table 17-4 lists the external RapidIO signals implemented on this interface.

Table 17-4. 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

17.2.2 Detailed Signal Descriptions

Table 17-5 describes the RapidIO interface signals in detail.

Table 17-5. 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 *MPC8540 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 17.1.4.1, “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 17-5. 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.

17.3 Memory Map/Register Definition

The memory map for the RapidIO controller is shown in Table 17-6.

Table 17-6. RapidIO Module Memory Map

Offset	Register	Access	Reset	Section/Page
RapidIO Architectural Registers				
0xC_0000	DIDCAR—Device identity capability register	R	0x0002_0002	17.3.1.1/17-13
0xC_0004	DICAR—Device information capability register	R	0x8030_0020	17.3.1.2/17-14
0xC_0008	AIDCAR—Assembly identity capability register	R/W	0x0000_0000	17.3.1.3/17-14
0xC_000C	AICAR—Assembly information capability register	R/W	0x0000_0100	17.3.1.4/17-15
0xC_0010	PEFCAR—Processing element features capability register	R	0xE088_0009	17.3.1.5/17-15
0xC_0014	SPICAR—Switch port information capability register	R	0x0000_0100	17.3.1.6/17-17
0xC_0018	SOCAR—Source operations capability register	R	0x0600_FCF0	17.3.1.7/17-17
0xC_001C	DOCAR—Destination operations capability register	R	0x0600_FCF4	17.3.1.8/17-19
0xC_0040	MSR—Mailbox command and status register	R	0x0000_0000	17.3.1.9/17-22
0xC_0044	PWDCSR—Port-write and doorbell command and status register	R	0x0000_0020	17.3.1.10/17-22
0xC_004C	PELLCCSR—Processing element logical layer control command and status register	R	0x0000_0001	17.3.1.11/17-24
0xC_0058	Reserved	R	0x0000_0000	—
0xC_005C	LCSBA1CSR—Local configuration space base address 1 command and status register	R/W	0x0000_0000	17.3.1.12/17-24
0xC_0060	BDIDCSR—Base device ID command and status register	R/W	0x00*nn*_0000	17.3.1.13/17-25
0xC_0068	HBDIDLCSR—Host base device ID lock command and status register	R/W	0x0000_FFFF	17.3.1.14/17-26
0xC_006C	CTCSR—Component tag command and status register	R/W	0x0000_0000	17.3.1.15/17-26
0xC_0100	PMBH0CSR—8/16 LP-LVDS port maintenance block header 0 command and status register	R	0x0000_0002	17.3.1.16/17-27

Table 17-6. 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	17.3.1.17/17-27
0xC_0124	PRTOCCSR—Port response time-out control command and status register	R/W	0xFFFF_FF00	17.3.1.18/17-28
0xC_013C	PGCCSR—Port general control command and status register	R/W	0x*n*000_0000	17.3.1.19/17-29
0xC_0140	PLMREQCSR—Port link maintenance request command and status register	R/W	0x0000_0000	17.3.1.20/17-29
0xC_0144	PLMRESPCSR—Port link maintenance response command and status register	R	0x0000_0000	17.3.1.21/17-30
0xC_0148	PLASCSR—Port local ackID status command and status register	R/W	0x0000_0000	17.3.1.22/17-31
0xC_0158	PESCSR—Port error and status command and status register	R/W	0x0000_0000	17.3.1.23/17-31
0xC_015C	PCCSR—Port control command and status register	R/W	0x4400_0000	17.3.1.24/17-33
RapidIO Implementation Registers				
0xD_0000	CR—Configuration register	R/W	0x0000_0001	17.3.2.1.1/17-34
0xD_0010	PCR—Port configuration register	R/W	0x0000_0010	17.3.2.1.2/17-35
0xD_0014	PEIR—Port error injection register	R/W	0x0000_0000	17.3.2.1.3/17-35
RapidIO ATMU Registers				
0xD_0C00	ROWTAR0—RapidIO outbound window translation address register 0	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0C10	ROWAR0—RapidIO outbound window attributes register 0	R/W	0x8004_401F	17.3.2.2.3/17-39
0xD_0C20	ROWTAR1—RapidIO outbound window translation address register 1	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0C28	ROWBAR1—RapidIO outbound window base address register 1	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0C30	ROWAR1—RapidIO outbound window attributes register 1	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0C40	ROWTAR2—RapidIO outbound window translation address register 2	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0C48	ROWBAR2—RapidIO outbound window base address register 2	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0C50	ROWAR2—RapidIO outbound window attributes register 2	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0C60	ROWTAR3—RapidIO outbound window translation address register 3	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0C68	ROWBAR3—RapidIO outbound window base address register 3	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0C70	ROWAR3—RapidIO outbound window attributes register 3	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0C80	ROWTAR4—RapidIO outbound window translation address register 4	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0C88	ROWBAR4—RapidIO outbound window base address register 4	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0C90	ROWAR4—RapidIO outbound window attributes register 4	R/W	0x0004_401F	17.3.2.2.3/17-39

Table 17-6. 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	17.3.2.2.1/17-37
0xD_0CA8	ROWBAR5—RapidIO outbound window base address register 5	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0CB0	ROWAR5—RapidIO outbound window attributes register 5	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0CC0	ROWTAR6—RapidIO outbound window translation address register 6	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0CC8	ROWBAR6—RapidIO outbound window base address register 6	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0CD0	ROWAR6—RapidIO outbound window attributes register 6	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0CE0	ROWTAR7—RapidIO outbound window translation address register 7	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0CE8	ROWBAR7—RapidIO outbound window base address register 7	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0CF0	ROWAR7—RapidIO outbound window attributes register 7	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0D00	ROWTAR8—RapidIO outbound window translation address register 8	R/W	0x0000_0000	17.3.2.2.1/17-37
0xD_0D08	ROWBAR8—RapidIO outbound window base address register 8	R/W	0x0000_0000	17.3.2.2.2/17-38
0xD_0D10	ROWAR8—RapidIO outbound window attributes register 8	R/W	0x0004_401F	17.3.2.2.3/17-39
0xD_0D60	RIWTAR4—RapidIO inbound window translation address register 4	R/W	0x0000_0000	17.3.2.2.4/17-40
0xD_0D68	RIWBAR4—RapidIO inbound window base address register 4	R/W	0x0000_0000	17.3.2.2.5/17-41
0xD_0D70	RIWAR4—RapidIO inbound window attributes register 4	R/W	0x0004_401F	17.3.2.2.6/17-42
0xD_0D80	RIWTAR3—RapidIO inbound window translation address register 3	R/W	0x0000_0000	17.3.2.2.4/17-40
0xD_0D88	RIWBAR3—RapidIO inbound window base address register 3	R/W	0x0000_0000	17.3.2.2.5/17-41
0xD_0D90	RIWAR3—RapidIO inbound window attributes register 3	R/W	0x0004_401F	17.3.2.2.6/17-42
0xD_0DA0	RIWTAR2—RapidIO inbound window translation address register 2	R/W	0x0000_0000	17.3.2.2.4/17-40
0xD_0DA8	RIWBAR2—RapidIO inbound window base address register 2	R/W	0x0000_0000	17.3.2.2.5/17-41
0xD_0DB0	RIWAR2—RapidIO inbound window attributes register 2	R/W	0x0004_401F	17.3.2.2.6/17-42
0xD_0DC0	RIWTAR1—RapidIO inbound window translation address register 1	R/W	0x0000_0000	17.3.2.2.4/17-40
0xD_0DC8	RIWBAR1—RapidIO inbound window base address register 1	R/W	0x0000_0000	17.3.2.2.5/17-41
0xD_0DD0	RIWAR1—RapidIO inbound window attributes register 1	R/W	0x0004_401F	17.3.2.2.6/17-42
0xD_0DE0	RIWTAR0—RapidIO inbound window translation address register 0	R/W	0x0000_0000	17.3.2.2.4/17-40
0xD_0DF0	RIWAR0—RapidIO inbound window attributes register 0	R/W	0x8004_401F	17.3.2.2.6/17-42

Table 17-6. 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	17.3.2.3.1/17-44
0xD_0E04	PNFEDiR—Port notification/fatal error detect disable register	R/W	0x0000_0000	17.3.2.3.2/17-47
0xD_0E08	PNFEIER—Port notification/fatal error interrupt enable register	R/W	0x0000_0000	17.3.2.3.3/17-49
0xD_0E0C	PECSR—Port error capture status register	R/W	0x0000_0000	17.3.2.3.4/17-52
0xD_0E10	EPCR0—Error packet capture register 0	R/W	0x0000_0000	17.3.2.3.5/17-52
0xD_0E14	EPCR1—Error packet capture register 1	R/W	0x0000_0000	17.3.2.3.6/17-53
0xD_0E18	EPCR2—Error packet capture register 2	R/W	0x0000_0000	17.3.2.3.16/17-58
0xD_0E20	PREDR—Port recoverable error detect register	R/W	0x0000_0000	17.3.2.3.23/17-61
0xD_0E28	PERTR—Port error recovery threshold register	R/W	0x00FF_0000	17.3.2.3.24/17-64
0xD_0E2C	PRTR—Port retry threshold register	R/W	0x00FF_0000	17.3.2.3.25/17-64
RapidIO Message Unit				
RapidIO Outbound Message Registers				
0xD_1000	OMR—Outbound mode register	R/W	0x0000_0000	17.3.3.1.1/17-65
0xD_1004	OSR—Outbound status register	R/W	0x0000_0000	17.3.3.1.2/17-67
0xD_100C	ODQDPAR—Outbound descriptor queue dequeue pointer address register	R/W	0x0000_0000	17.3.3.1.3/17-68
0xD_1014	OSAR—Outbound source address register	R/W	0x0000_0000	17.3.3.1.4/17-69
0xD_1018	ODPR—Outbound destination port register	R/W	0x0000_0000	17.3.3.1.5/17-70
0xD_101C	ODATR—Outbound destination attributes register	R/W	0x0006_0000	17.3.3.1.6/17-70
0xD_1020	ODCR—Outbound double-word count register	R/W	0x0000_0000	17.3.3.1.7/17-71
0xD_1028	ODQEPAR—Outbound descriptor queue enqueue pointer address register	R/W	0x0000_0000	17.3.3.1.8/17-72
RapidIO Inbound Message Registers				
0xD_1060	IMR—Inbound mailbox mode register	R/W	0x0000_0000	17.3.3.2.1/17-72
0xD_1064	ISR—Inbound mailbox status register	R/W	0x0000_0000	17.3.3.2.2/17-74
0xD_106C	IFQDPAR—Inbound frame queue dequeue pointer address register	R/W	0x0000_0000	17.3.3.2.3/17-75
0xD_1074	IFQEPAR—Inbound frame queue enqueue pointer address register	R/W	0x0000_0000	17.3.3.2.4/17-76
RapidIO Doorbell Registers				
0xD_1460	DMR—Doorbell mode register	R/W	0x0000_0000	17.3.3.3.1/17-77
0xD_1464	DSR—Doorbell status register	R/W	0x0000_0000	17.3.3.3.2/17-78

Table 17-6. RapidIO Module Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0xD_146C	DQDPAR—Doorbell queue dequeue pointer address register	R/W	0x0000_0000	17.3.3.3.3/17-79
0xD_1474	DQEPAR—Doorbell queue enqueue pointer address register	R/W	0x0000_0000	17.3.3.3.4/17-80
RapidIO Port-Write Registers				
0xD_14E0	PWMR—Port-write mode register	R/W	0x0000_0000	17.3.3.4.1/17-81
0xD_14E4	PWSR—Port-write status register	R/W	0x0000_0000	17.3.3.4.2/17-82
0xD_14EC	PWQBAR—Port-write queue base address register	R/W	0x0000_0000	17.3.3.4.3/17-83

17.3.1 Architectural Registers

Following are the architectural registers of the RapidIO module. For additional details of these registers, refer to the *Parallel RapidIO Interconnect Specification*.

17.3.1.1 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 17-3 describes the DIDCAR.

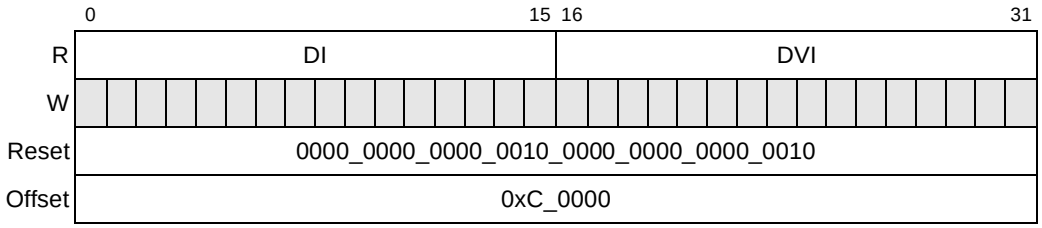


Figure 17-3. Device Identity Capability Register (DIDCAR)

Table 17-7 describes the fields of the DIDCAR.

Table 17-7. 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. 0x0002—MPC8540
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

17.3.1.2 Device Information Capability Register (DICAR)

The DICAR, shown in Figure 17-4, 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).

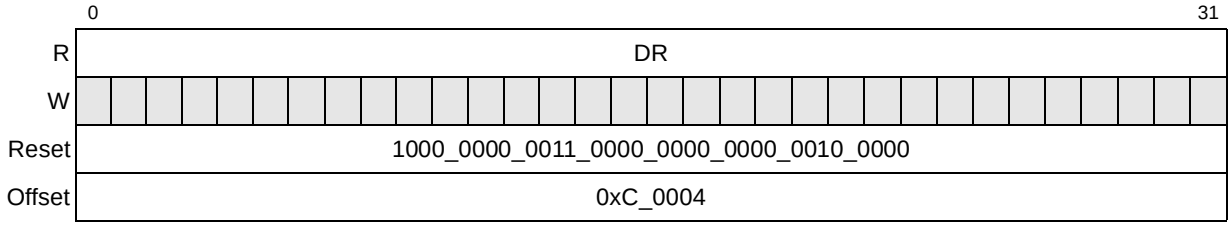


Figure 17-4. Device Information Capability Register (DICAR)

Table 17-8 describes the field of the DICAR.

Table 17-8. DICAR Field Description

Bits	Name	Description
0–31	DR	Device revision—Copy of the device's system version register (SVR). Draco: 0x8030_0020

17.3.1.3 Assembly Identity Capability Register (AIDCAR)

AIDCAR, shown in Figure 17-5, provides information regarding the type of assembly or subsystem being used as well as the vendor who manufactured it.

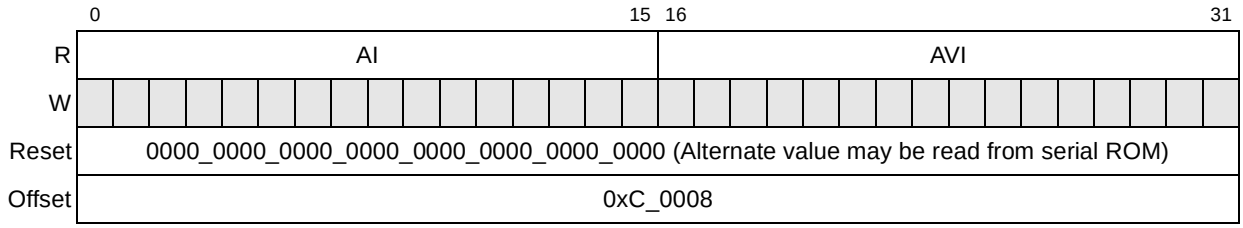


Figure 17-5. Assembly Identity Capability Register (AIDCAR)

Table 17-9 describes the fields of the AIDCAR.

Table 17-9. 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.

17.3.1.4 Assembly Information Capability Register (AICAR)

The AICAR, shown in Figure 17-6, provides additional information about the assembly.

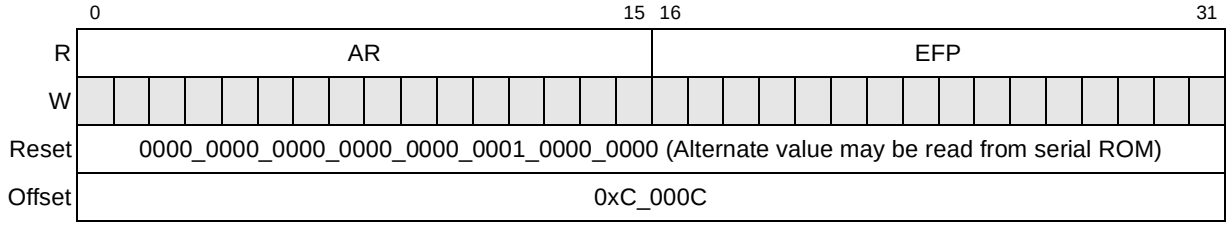


Figure 17-6. Assembly Information Capability Register (AICAR)

Table 17-10 describes the fields of the AICAR.

Table 17-10. 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.

17.3.1.5 Processing Element Features Capability Register (PEFCAR)

The PEFCAR, shown in Figure 17-7, identifies major functionality provided by the RapidIO block.

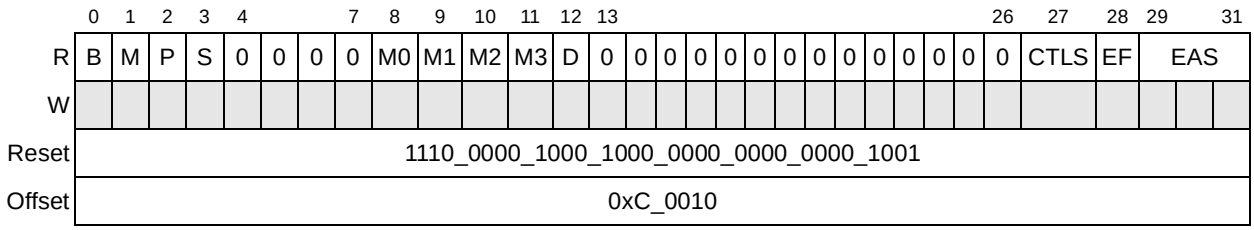


Figure 17-7. Processing Element Features Capability Register (PEFCAR)

Table 17-11 describes the fields of the PEFCAR.

Table 17-11. PEFCAR Field Descriptions

Bits	Name	Description
0	B	Bridge. Set for the MPC8540. 0 The RapidIO controller does not have a DRAM interface. 1 The RapidIO controller has a DRAM interface.
1	M	Memory. Set for the MPC8540. 0 The RapidIO controller does not have local address space. 1 The RapidIO controller has local address space.

Table 17-11. PEFCAR Field Descriptions

Bits	Name	Description
2	P	Processor. Set for the MPC8540. 0 The RapidIO controller does not have a processor core. 1 The RapidIO controller has a processor core.
3	S	Switch. Cleared for the MPC8540. 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 MPC8540. 0 The RapidIO controller does not support mailbox 0. 1 The RapidIO controller supports mailbox 0.
9	M1	Mailbox 1. Cleared for the MPC8540. 0 The RapidIO controller does not support mailbox 1. 1 The RapidIO controller supports mailbox 1.
10	M2	Mailbox 2. Cleared for the MPC8540. 0 The RapidIO controller does not support mailbox 2. 1 The RapidIO controller supports mailbox 2.
11	M3	Mailbox 3. Cleared for the MPC8540. 0 The RapidIO controller does not support mailbox 3. 1 The RapidIO controller supports mailbox 3.
12	D	Doorbell. Set for the MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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.

17.3.1.6 Switch Port Information Capability Register (SPICAR)

The SPICAR, shown in Figure 17-8, defines the RapidIO block's switching capabilities.

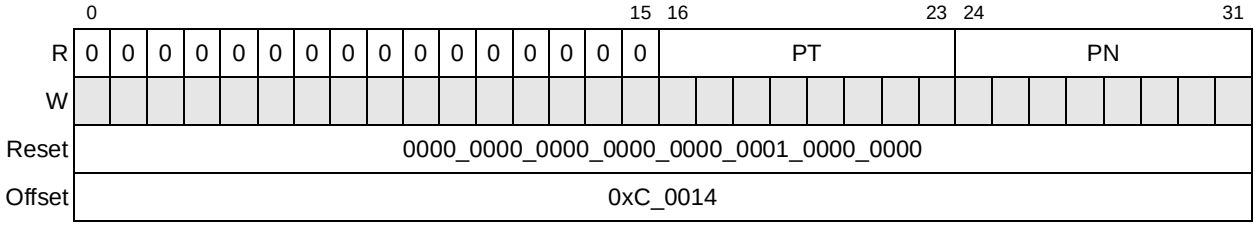


Figure 17-8. Switch Port Information Capability Register (SPICAR)

Table 17-12 describes the fields of the SPICAR.

Table 17-12. 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 MPC8540 has only one port.

17.3.1.7 Source Operations Capability Register (SOCAR)

The SOCAR, shown in Figure 17-9, reflects the set of RapidIO operations that this device can initiate. See the *Parallel RapidIO Interconnect Specification* for details.

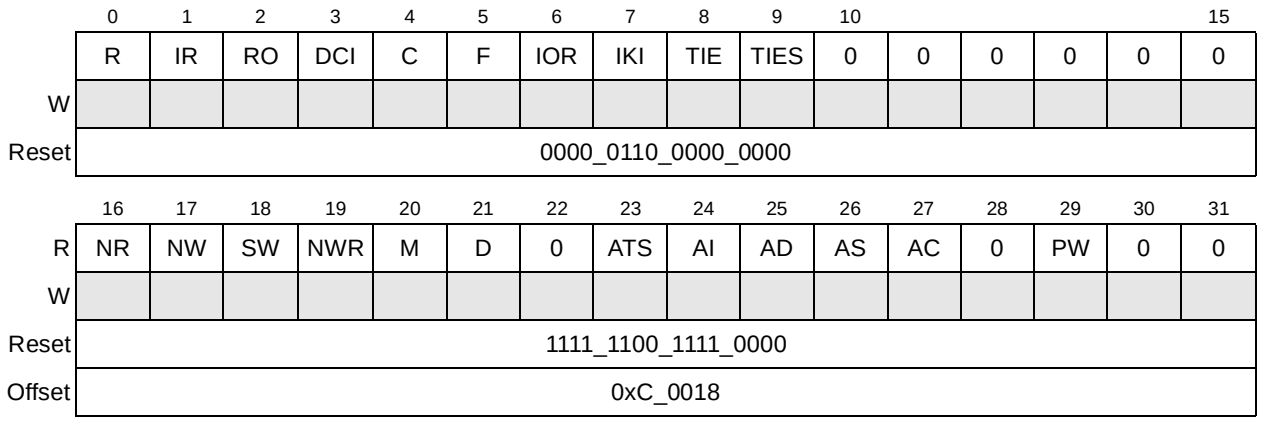


Figure 17-9. Source Operations Capability Register (SOCAR)

Table 17-13 describes the fields of the SOCAR.

Table 17-13. SOCAR Field Descriptions

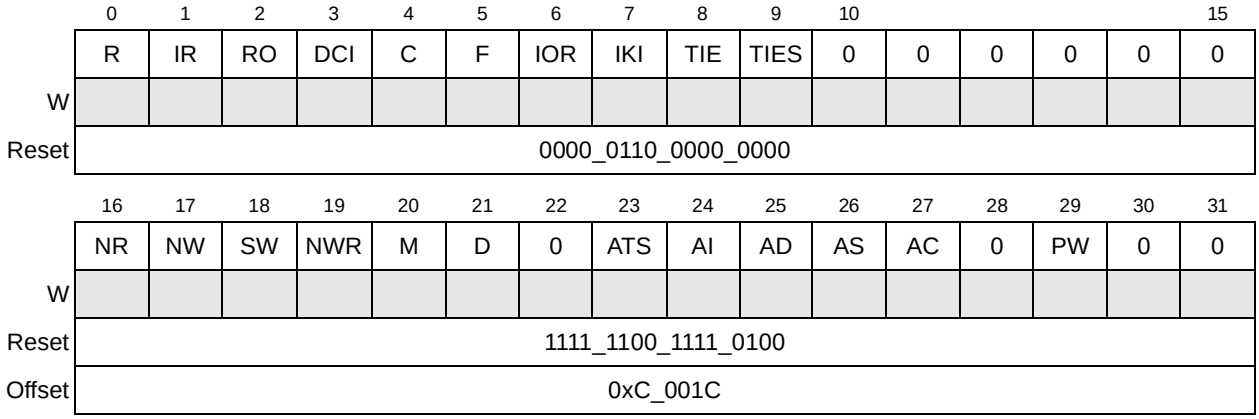
Bits	Name	Description
0	R	Read operation. Cleared for the MPC8540. 0 The RapidIO controller cannot initiate a read operation. 1 The RapidIO controller can initiate a read operation.
1	IR	Iread operation. Cleared for the MPC8540. 0 The RapidIO controller cannot initiate an iread operation. 1 The RapidIO controller can initiate an iread operation.
2	RO	Read to own operation. Cleared for the MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 0 The RapidIO controller cannot initiate an swrite operation. 1 The RapidIO controller can initiate an swrite operation.

Table 17-13. SOCAR Field Descriptions (continued)

Bits	Name	Description
19	NWR	Nwrite_r operation. (Write non-sharable memory.) Set for the MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 0 The RapidIO controller cannot initiate a port-write operation. 1 The RapidIO controller can initiate a port-write operation.
30–31	—	Reserved

17.3.1.8 Destination Operations Capability Register (DOCAR)

The DOCAR, shown in Figure 17-10, describes RapidIO I/O operations that this device can service. See the *Parallel RapidIO Interconnect 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 17-10. Destination Operations Capability Register (DOCAR)

Table 17-14 describes the fields of the DOCAR.

Table 17-14. DOCAR Field Descriptions

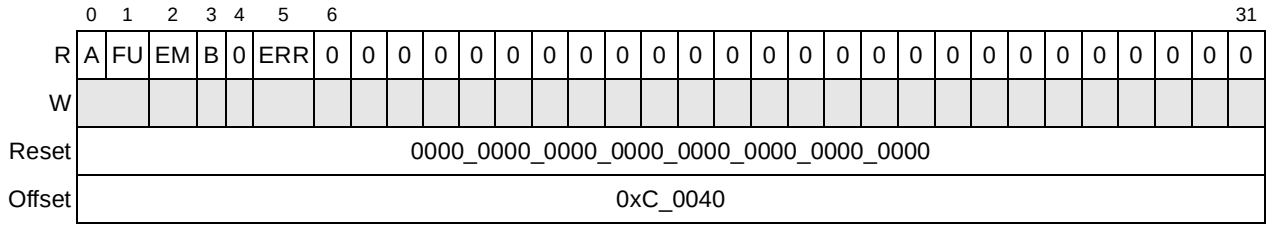
Bits	Name	Description
0	R	Read operation. Cleared for the MPC8540. 0 The RapidIO controller cannot service a read operation. 1 The RapidIO controller can service a read operation.
1	IR	Iread operation. Cleared for the MPC8540. 0 The RapidIO controller cannot service an iread operation. 1 The RapidIO controller can service an iread operation.
2	RO	Read to own operation. Cleared for the MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 0 The RapidIO controller cannot service a TLBIE operation. 1 The RapidIO controller can service a TLBIE operation.

Table 17-14. DOCAR Field Descriptions (continued)

Bits	Name	Description
9	TIES	TLBSYNC operation. Cleared for the MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 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 MPC8540. 0 The RapidIO controller cannot service a port-write operation. 1 The RapidIO controller can service a port-write operation.
30–31	—	Reserved

17.3.1.9 Mailbox Command and Status Register (MSR)

The MSR, shown in Figure 17-11, reflects the status of the RapidIO mailbox controller on this device. Note that the RapidIO controller only supports Mailbox 0.



	0	1	2	3	4	5	6																							31
R	A	FU	EM	B	0	ERR	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	0xC_0040																													

Figure 17-11. Mailbox Status Register (MSR)

Table 17-15 describes the fields of the MSR.

Table 17-15. 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

17.3.1.10 Port-Write and Doorbell Command and Status Register (PWDCSR)

The PWDCSR, shown in Figure 17-12, reflects the status of the RapidIO doorbell and port-write hardware on this device. Additional details can be found in the *Parallel RapidIO Interconnect Specification* in the sections titled “Doorbell CSR” and “Write Port CSR.”

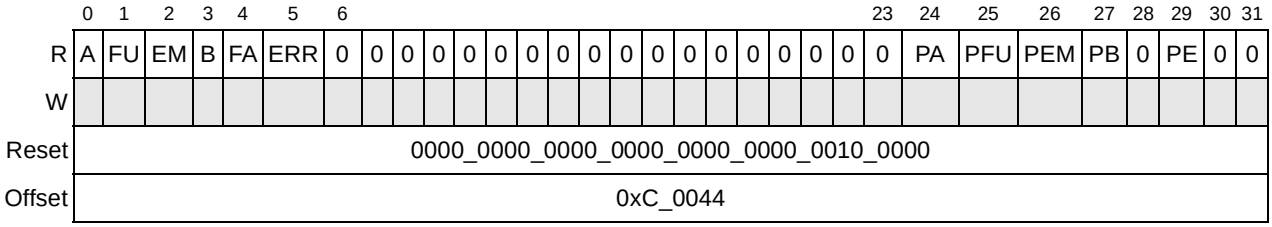


Figure 17-12. Port-Write and Doorbell Command and Status Register (PWDCSR)

Table 17-16 describes the fields of the PWDCSR.

Table 17-16. 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 MPC8540.) 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 17-16. 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

17.3.1.11 Processing Element Logic Layer Control Command and Status Register (PELLCCSR)

The PELLCCSR, shown in Figure 17-13, is a read-only register. Additional details of this register can be found in the *Parallel RapidIO Interconnect 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	EAC		
W																														
Reset	0000_0000_0000_0000_0000_0000_0000_0001																													
Offset	0xC_004C																													

Figure 17-13. Processing Element Logic Layer Control Command and Status Register (PELLCCSR)

Table 17-17 describes the fields of the PELLCCSR.

Table 17-17. PELLCCSR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29–31	EAC	Extended addressing control. The RapidIO controller supports 34-bit addresses (read-only).

17.3.1.12 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 17-14, contains the 14 msbs of the double-word address, which represents the base address for the device's register space.

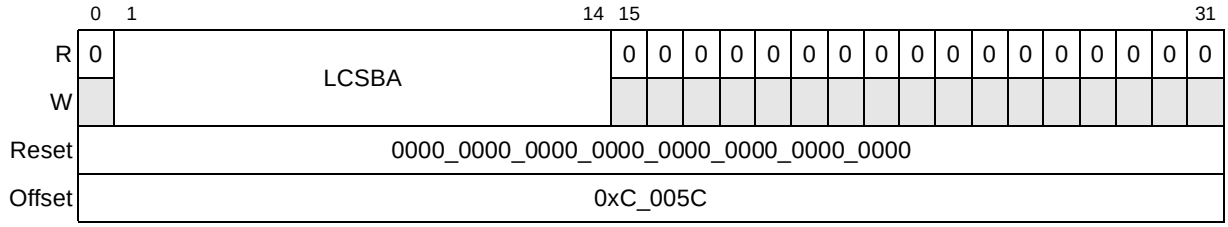


Figure 17-14. Local Configuration Space Base Address 1 Command and Status Register (LCSBA1CSR)

Table 17-18 describes the fields of the LCSBA1CSR.

Table 17-18. 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

17.3.1.13 Base Device ID Command and Status Register (BDIDCSR)

Figure 17-15 shows the BDIDSCR.

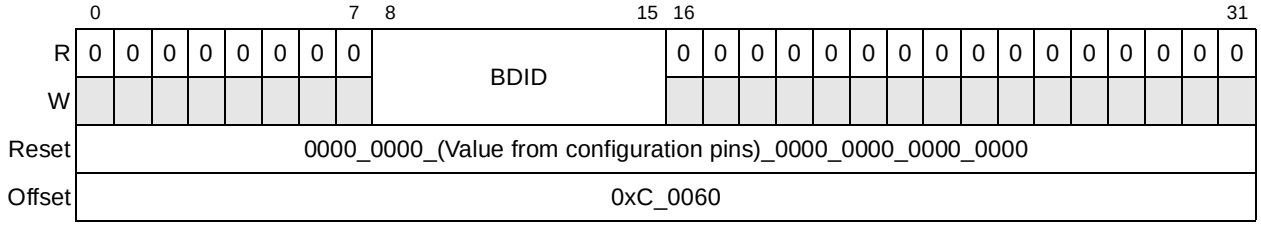


Figure 17-15. Base Device ID Command and Status Register (BDIDCSR)

Table 17-19 describes the fields of the BDIDCSR.

Table 17-19. 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

17.3.1.14 Host Base Device ID Lock Command and Status Register (HBDIDLCSR)

The host base device ID lock CSR, shown in Figure 17-16, 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.

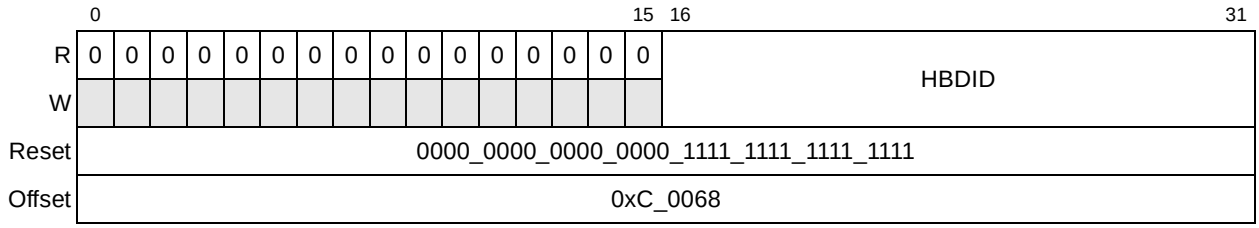


Figure 17-16. Host Base Device ID Lock Command and Status Register (HBDIDLCSR)

Table 17-20 describes the fields of the HBDIDLCSR.

Table 17-20. 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.

17.3.1.15 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 17-17 shows the CTCSR.

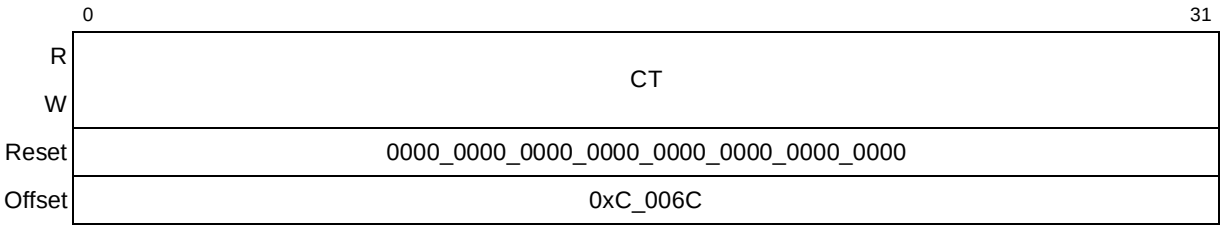


Figure 17-17. Component Tag Command and Status Register (CTCSR)

Table 17-21 describes the field of the CTCSR.

Table 17-21. CTCSR Field Descriptions

Bits	Name	Description
0–31	CT	Component tag. Identifying component tag for the RapidIO block.

17.3.1.16 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 17-18, 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.

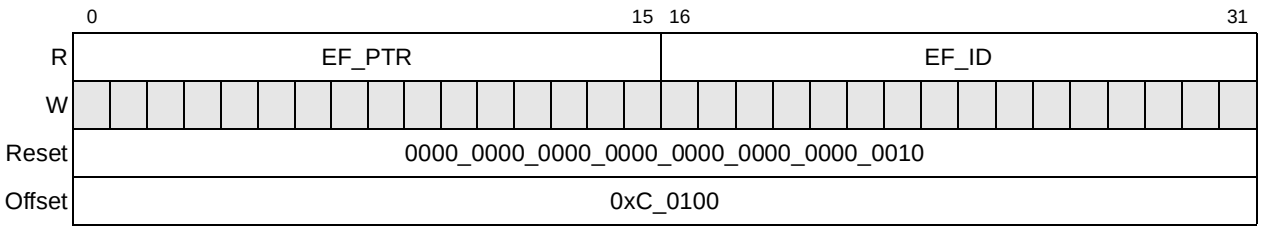


Figure 17-18. 8/16 LP-LVDS Port Maintenance Block Header 0 Command and Status Register (PMBH0CSR)

Table 17-22 describes the fields of the PMBH0CSR.

Table 17-22. PMBH0CSR Field Descriptions

Bits	Name	Description
0–15	EF_PTR	Extended features pointer. Points to the next extended features block. 0x0000 for the MPC8540.
16–31	EF_ID	Extended features ID. Identifies this as the generic end point port maintenance block header. 0x0002 for the MPC8540.

17.3.1.17 Port Link Time-Out Control Command and Status Register (PLTOCCSR)

The PLTOCCSR, shown in Figure 17-19, 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.

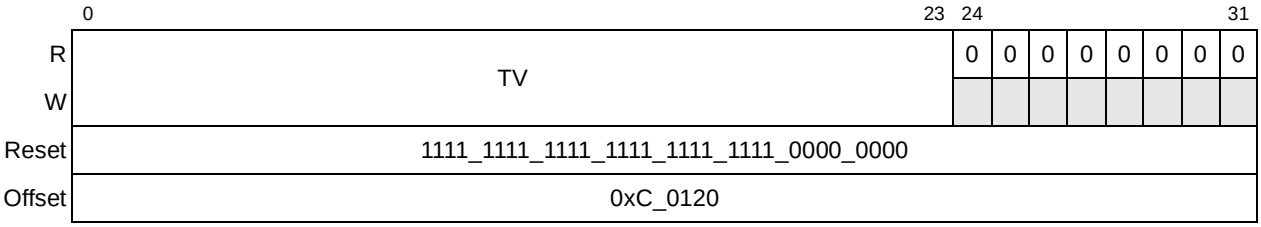


Figure 17-19. Port Link Time-Out Control Command and Status Register (PLTOCCSR)

Table 17-23 describes the fields of the PLTOCCSR.

Table 17-23. PLTOCCSR Field Descriptions

Bits	Name	Description
0–23	TV	Time-out value. This value is expressed in platform clock cycles.
24–31	—	Reserved

17.3.1.18 Port Response Time-Out Control Command and Status Register (PRTOCCSR)

PRTOCCSR, shown in Figure 17-20, contains the time-out timer value used to monitor port events, for instance, between sending a request packet and receiving the corresponding response packet.

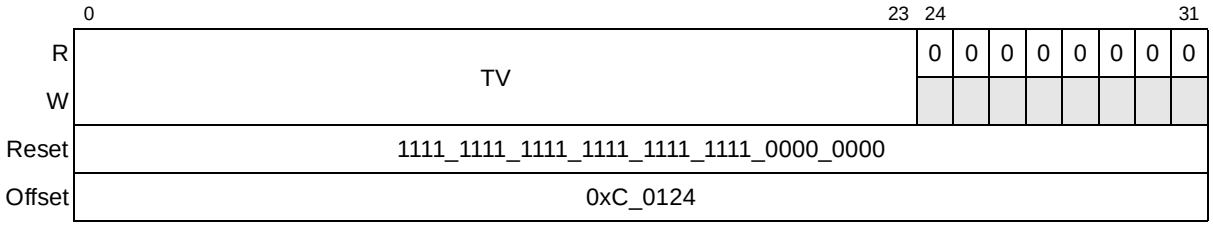


Figure 17-20. Port Response Time-Out Control Command and Status Register (PRTOCCSR)

Table 17-24 describes the fields of the PRTOCCSR.

Table 17-24. PRTOCCSR Field Descriptions

Bits	Name	Description
0–23	TV	Time-out value. This value is expressed in platform clock cycles.
24–31	—	Reserved

17.3.1.19 Port General Control Command and Status Register (PGCCSR)

The PGCCSR, shown in Figure 17-21, 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 17-21. Port General Control Command and Status Register (PGCCSR)

Table 17-25 describes the fields of the PGCCSR.

Table 17-25. 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

17.3.1.20 Port Link Maintenance Request Command and Status Register (PLMREQCSR)

A write to the PLMREQCSR, shown in Figure 17-22, generates a link-request control symbol on the RapidIO port interface. See the *Parallel RapidIO Interconnect Specification* for descriptions of the reset command, input-status command and send-training commands.

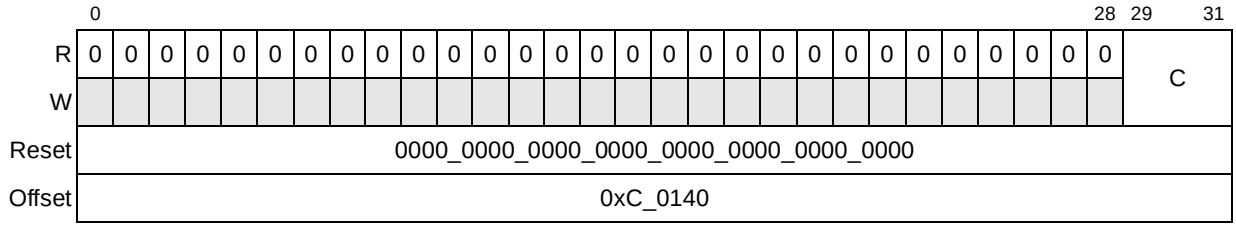


Figure 17-22. Port Link Maintenance Request Command and Status Register (PLMREQCSR)

Table 17-26 describes the fields of the PLMREQCSR.

Table 17-26. 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 17-85 for command symbol formats.

17.3.1.21 Port Link Maintenance Response Command and Status Register (PLMRESPCSR)

A read from the PLMRESPCSR, shown in Figure 17-23, returns the status received in a link-response control symbol.

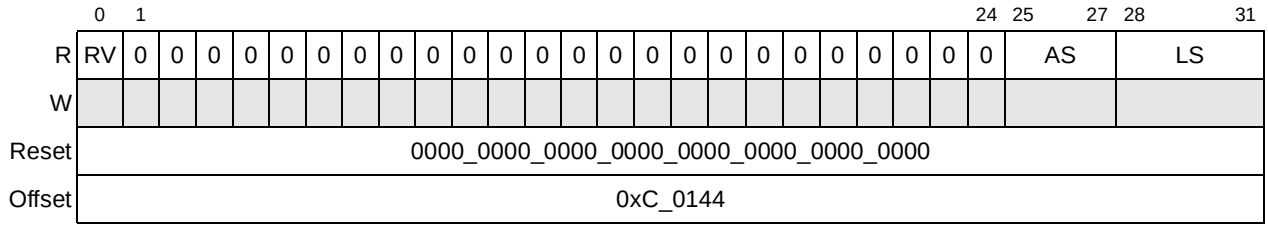


Figure 17-23. Port Link Maintenance Response Command and Status Register (PLMRESPCSR)

Table 17-27 describes the fields of the PLMRESPCSR.

Table 17-27. 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 17-27. 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.

17.3.1.22 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 17-24 shows the PLASCSR.

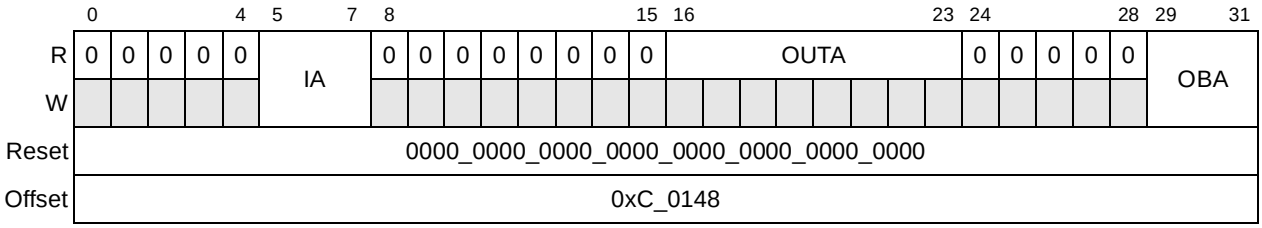


Figure 17-24. Port Local AckID Status Command and Status Register (PLASCSR)

Table 17-28 describes the fields of the PLASCSR.

Table 17-28. 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.

17.3.1.23 Port Error and Status Command and Status Register (PESCSR)

The PESCSR contains port error and status information. Figure 17-25 shows the PESCSR.

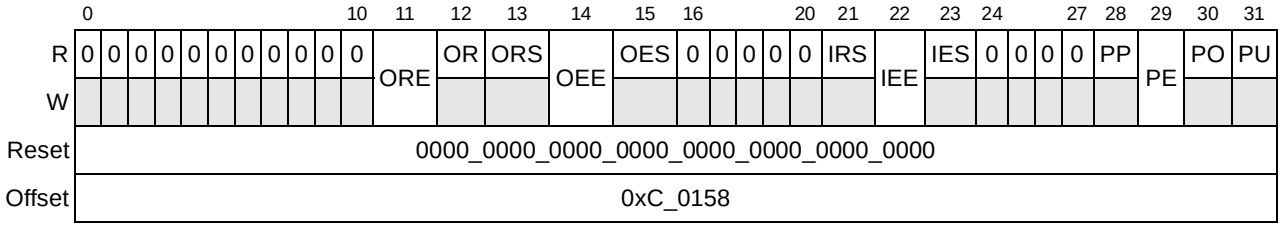


Figure 17-25. Port Error and Status Command and Status Register (PESCSR)

Table 17-29 describes the fields of the PESCSR.

Table 17-29. 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
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.)

Table 17-29. PESCSR Field Descriptions (continued)

Bits	Name	Description
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

17.3.1.24 Port Control Command and Status Register (PCCSR)

The PCCSR contains bits used for port control. Figure 17-26 shows the PCCSR.

	0	1	2	3	4	5	6	7	8	9																					31
R	OPW	OPE	OPD	0	IPW	IPE	IPD	0	ECD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																															
Reset	0100_0100_0000_0000_0000_0000_0000_0000																														
Offset	0xC_015C																														

Figure 17-26. Port Control Command and Status Register (PCCSR)

Table 17-30 describes the fields of the PCCSR.

Table 17-30. 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
4	IPW	Operating width of the input port (read-only field) 0 Port operating in 8-bit mode 1 Reserved

Table 17-30. PCCSR Field Descriptions (continued)

Bits	Name	Description
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

17.3.2 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 Interconnect Specification*.

17.3.2.1 General Registers

17.3.2.1.1 Configuration Register (CR)

The CR contains configuration information regarding the behavior of the RapidIO controller. Figure 17-27 shows the CR.

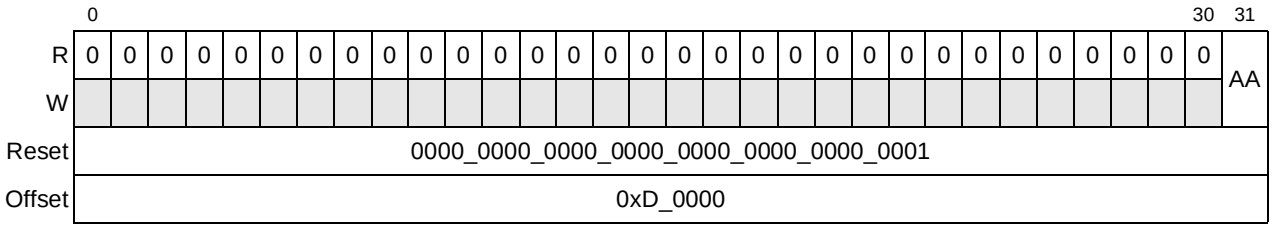


Figure 17-27. Configuration Register (CR)

Table 17-31 describes the fields of the CR.

Table 17-31. 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.

17.3.2.1.2 Port Configuration Register (PCR)

The PCR contains configuration information regarding port behavior of the RapidIO controller. Figure 17-28 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	0	0	CCE	0	0	0	0
W																															
Reset	0000_0000_0000_0000_0000_0000_0001_0000																														
Offset	0xD_0010																														

Figure 17-28. Port Configuration Register (PCR)

Table 17-32 describes the fields of the PCR.

Table 17-32. 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

17.3.2.1.3 Port Error Injection Register (PEIR)

RapidIO error injection registers, shown in Figure 17-29, 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.

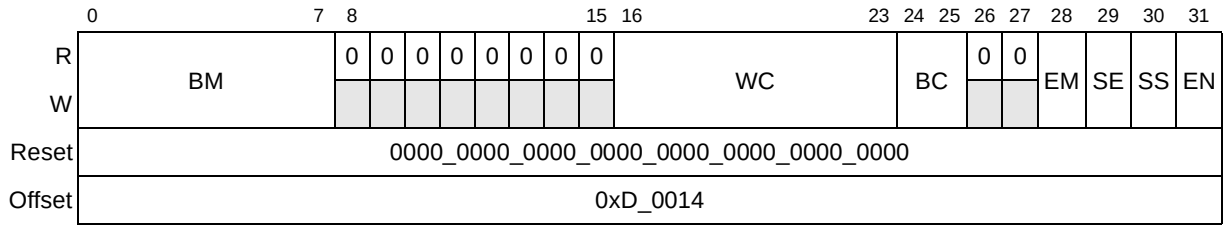


Figure 17-29. Port Error Injection Register (PEIR)

Table 17-33 describes the fields of the PEIR.

Table 17-33. 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.

17.3.2.2 ATMU Registers

Although the *Parallel RapidIO Interconnect Specification* allows for 48- and 64-bit addresses, the MPC8540 implementation only allows for 34-bit RapidIO addresses. ATMU window misses use the window 0 register set by default. Overlapping window hits result in the use of the lowest number window register set hit. For both inbound and outbound translation, the smallest window size is 4 Kbytes and the largest window size is 4 Gbytes. See Section 17.7, “ATMU (Address Translation and Mapping Unit),” for more information.

17.3.2.2.1 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 17.7.1, "Outbound ATMU Translation," for more information. Figure 17-30 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 17-30. RapidIO Outbound Window Translation Address Registers 0–8 for Standard Transactions (ROWTAR*n*)

Table 17-34 describes the fields of the ROWTARs for standard operations.

Table 17-34. 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 17-31 shows the ROWTARs for maintenance transactions.

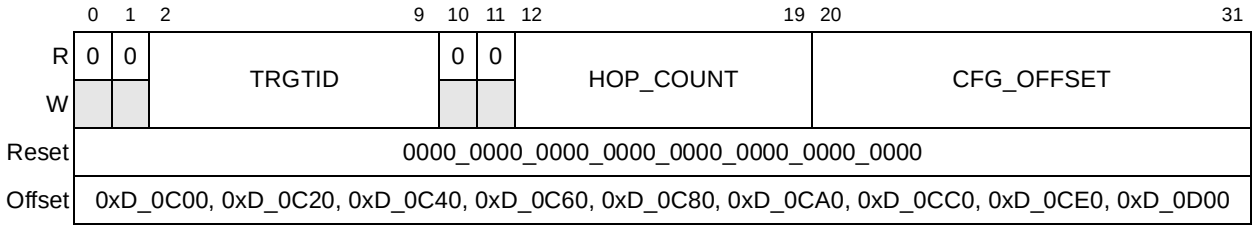


Figure 17-31. RapidIO Outbound Window Translation Address Registers 0–8 for Maintenance Transactions (ROWTAR*n*)

Table 17-35 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 17-35. 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.

17.3.2.2.2 RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR*n*)

The RapidIO outbound window translation address registers (ROWBAR*n*), shown in Figure 17-32, 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.

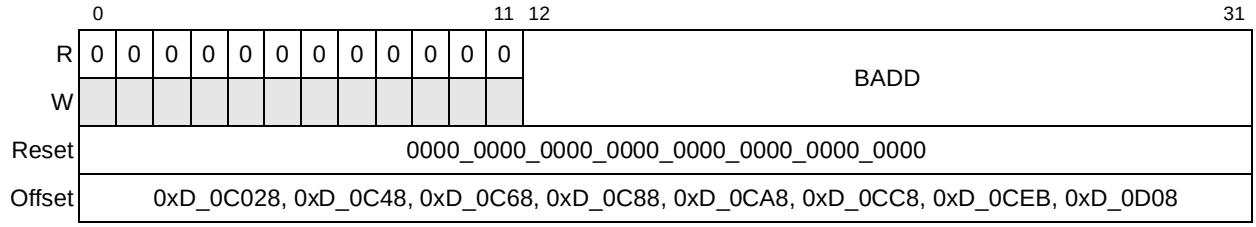


Figure 17-32. RapidIO Outbound Window Base Address Registers 1–8 (ROWBAR*n*)

Table 17-36 describes the fields of the ROWBARs.

Table 17-36. 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.

17.3.2.2.3 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 17-33 and Figure 17-34 show the ROWARs.

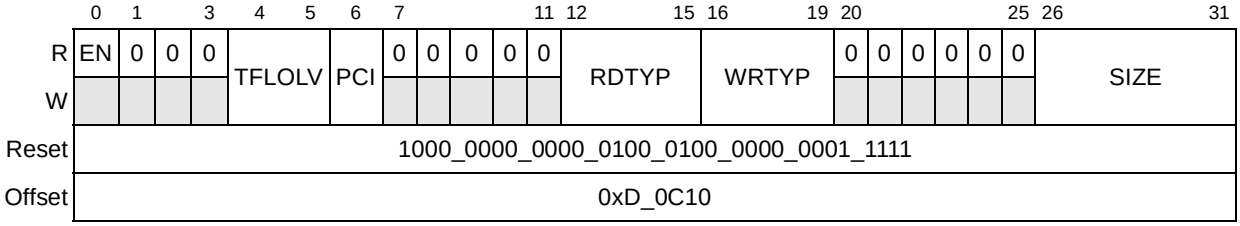


Figure 17-33. RapidIO Outbound Window Attributes Register 0 (ROWAR0)

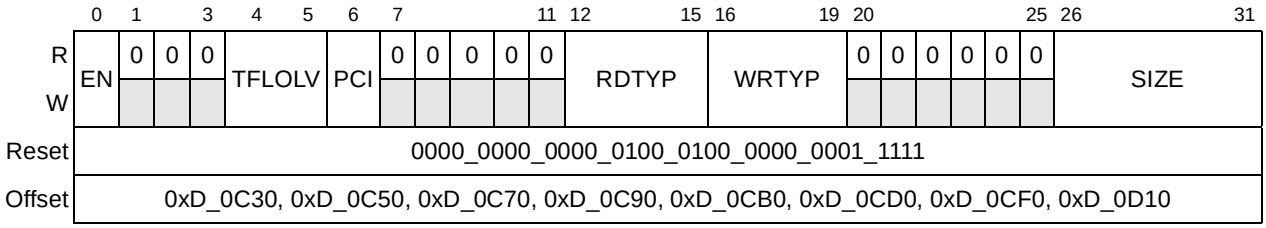


Figure 17-34. RapidIO Outbound Window Attributes Registers 1–8 (ROWAR*n*)

Table 17-37 describes the fields of the ROWARs.

Table 17-37. 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

Table 17-37. ROWAR*n* Field Descriptions (continued)

Bits	Name	Description
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 17.7.1.2, “Outbound Special Transactions and Requirements,” for more information on maintenance transactions. 0000–0001Reserved 0010 I/O read home 0011 Reserved 0100 nread 0101–0110Reserved 0111 Maintenance read 1000–1011Reserved 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 17.7.1.2, “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 1000–1111Reserved
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

17.3.2.2.4 RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR*n*)

The RapidIO inbound window translation address registers (RIWTARs), shown in Figure 17-35, 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 17.7.2, “Inbound ATMU Translation,” for more information.

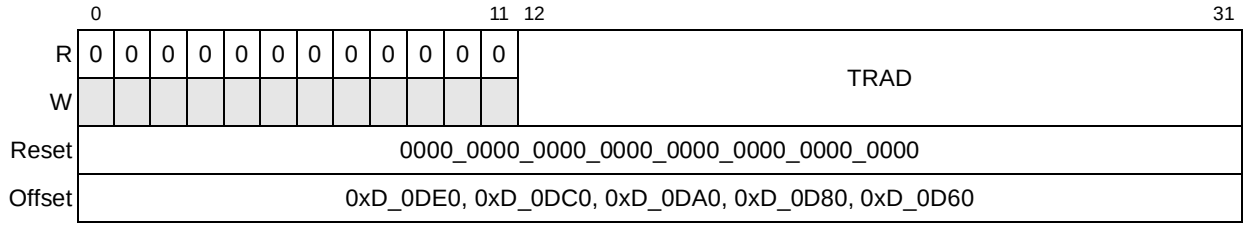


Figure 17-35. RapidIO Inbound Window Translation Address Registers 0–4 (RIWTAR*n*)

Table 17-38 describes the fields of the RIWTARs.

Table 17-38. 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.

17.3.2.2.5 RapidIO Inbound Window Base Address Registers 1–4 (RIWBAR*n*)

RapidIO inbound window translation address registers (RIWBARs), shown in Figure 17-36, 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.

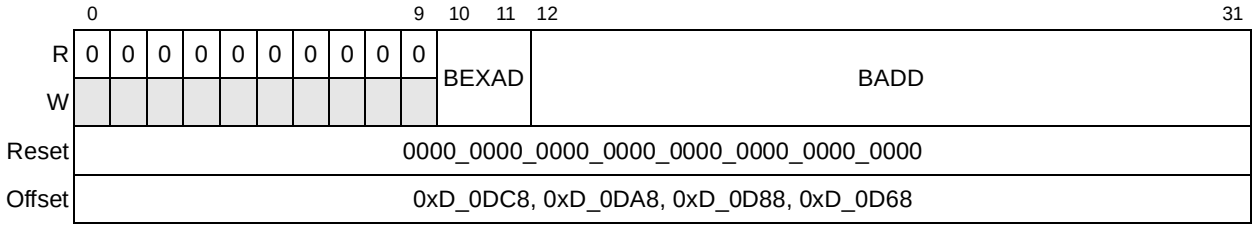


Figure 17-36. RapidIO Inbound Window Base Address Registers 1–4 (RIWBAR*n*)

Table 17-39 describes the fields of the RIWBARs.

Table 17-39. 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.

17.3.2.2.6 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 17-38 and Figure 17-37 show the RIWARs.

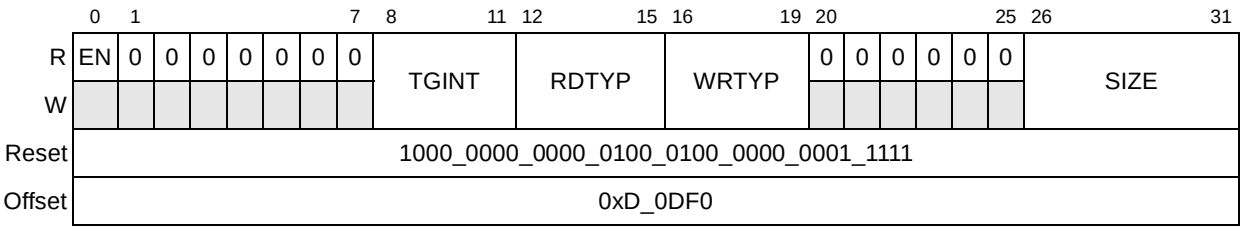


Figure 17-37. RapidIO Inbound Window Attributes Register 0 (RIWAR0)

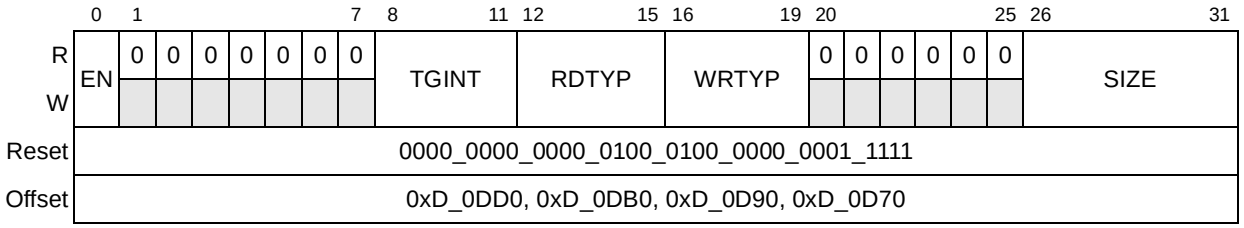


Figure 17-38. RapidIO Inbound Window Attributes Registers 1–4 (RIWAR*n*)

Table 17-40 describes the fields of the RIWARs.

Table 17-40. 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 17-40. 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

17.3.2.3 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]).

17.3.2.3.1 Port Notification/Fatal Error Detect Register (PNFEDR)

The PNFEDR, shown in Figure 17-39, 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 17-39. Port Notification/Fatal Error Detect Register (PNFEDR)

Table 17-41 describes the fields of the PNFEDR.

Table 17-41. 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 17-41. 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 17-41. 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)

17.3.2.3.2 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 17-40 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 17-40. Port Notification/Fatal Error Detect Disable Register (PNFEDiR)

Table 17-42 describes the fields of the PNFEDiR.

Table 17-42. 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 17-42. 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 17-42. 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

17.3.2.3.3 Port Notification/Fatal Error Interrupt Enable Register (PNFEIER)

The PNFEIER, shown in Figure 17-41, 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 17-41. Port Notification/Fatal Error Interrupt Enable Register (PNFEIER)

Table 17-43 describes the fields of the PNFEIER.

Table 17-43. 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 17-43. 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

17.3.2.3.4 Port Error Capture Status Register (PECSR)

The PECSR, shown in Figure 17-42, 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.

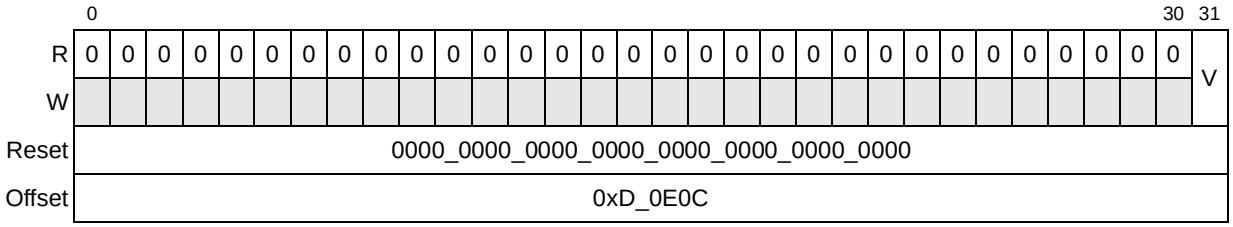


Figure 17-42. Port Error Capture Status Register (PECSR)

Table 17-44 describes the fields of the PECSR.

Table 17-44. 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.

17.3.2.3.5 Error Packet Capture Register 0 (EPCR0)

The EPCR0, shown in Figure 17-43, 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.

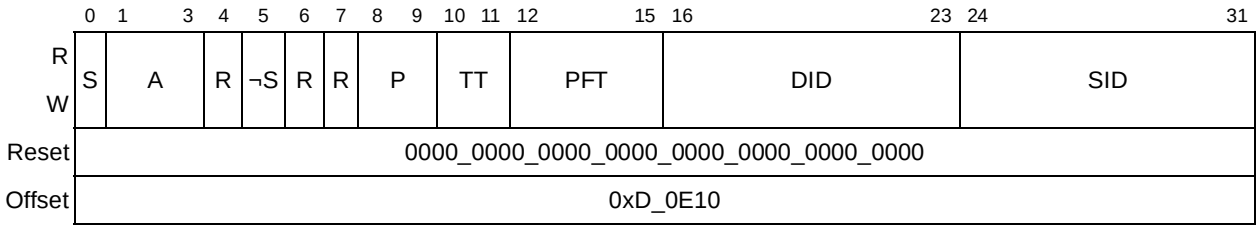


Figure 17-43. Error Packet Capture Register 0 (EPCR0)

Table 17-45 describes the fields of the EPCR0.

Table 17-45. 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

17.3.2.3.6 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 17.3.2.3.7, “EPCR1—Type 1 Packet Format,” through Section 17.3.2.3.15, “EPCR1—Type 13 Packet Format.”

17.3.2.3.7 EPCR1—Type 1 Packet Format

Figure 17-44 describes EPCR1 for a type 1 packet format (EPCR0[PFT] = 1).

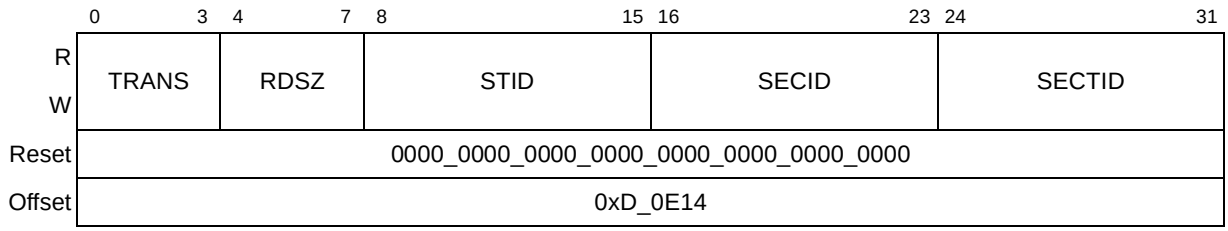


Figure 17-44. Error Packet Capture Register 1 (EPCR1)—Type 1 Packet Format

Table 17-46 describes the EPCR1 fields for a type 1 packet format (EPCR0[PFT] = 1).

Table 17-46. 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

17.3.2.3.8 EPCR1—Type 2 Packet Format

Figure 17-45 describes the EPCR1 for a type 2 packet format (EPCR0[PFT] = 2).

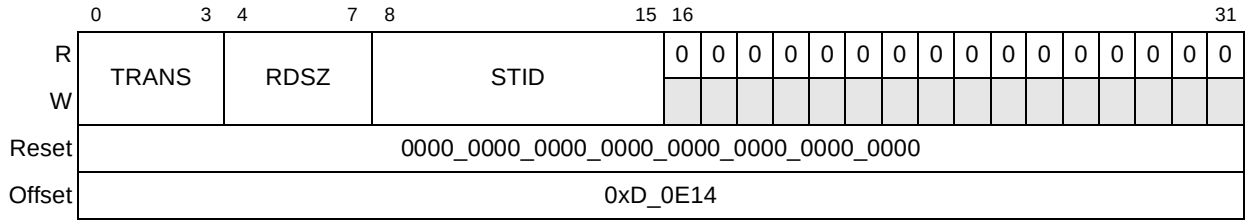


Figure 17-45. Error Packet Capture Register 1 (EPCR1)—Type 2 Packet Format

Table 17-47 describes the EPCR1 fields for a type 2 packet format (EPCR0[PFT] = 2).

Table 17-47. 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

17.3.2.3.9 EPCR1—Type 5 Packet Format

Figure 17-46 describes the EPCR1 for a type 5 packet format (EPCR0[PFT] = 5).

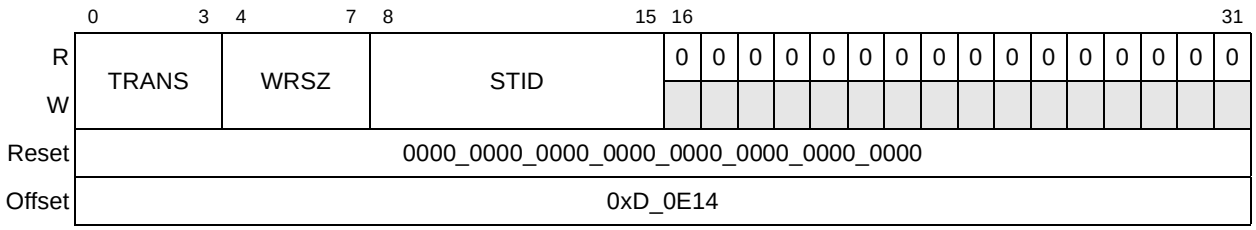


Figure 17-46. Error Packet Capture Register 1 (EPCR1)—Type 5 Packet Format

Table 17-48 describes the EPCR1 fields for a type 5 packet format (EPCR0[PFT] = 5).

Table 17-48. 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

17.3.2.3.10 EPCR1—Type 6 Packet Format

Figure 17-47 shows the EPCR1 for a type 6 packet format (EPCR0[PFT] = 6).

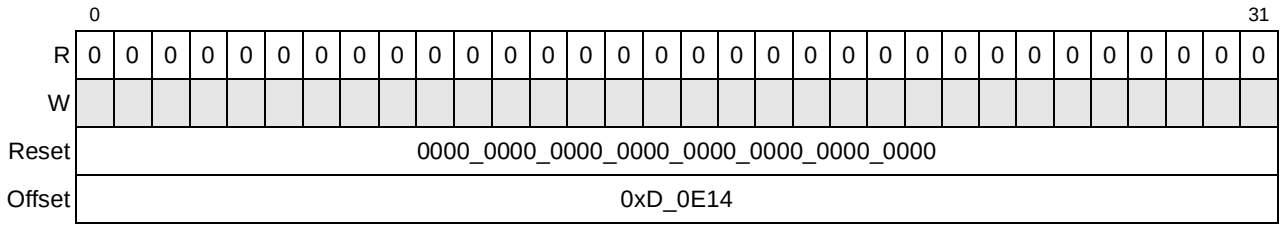


Figure 17-47. Error Packet Capture Register 1 (EPCR1)—Type 6 Packet Format

Table 17-49 describes the field of the EPCR1 for a type 6 packet format (EPCR0[PFT] = 6).

Table 17-49. EPCR1 Type 6 Field Descriptions

Bits	Name	Description
0–31	—	Reserved

17.3.2.3.11 EPCR1—Type 8 Request Packet Format

Figure 17-48 shows EPCR1 for a type 8 request packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 0 or 1).

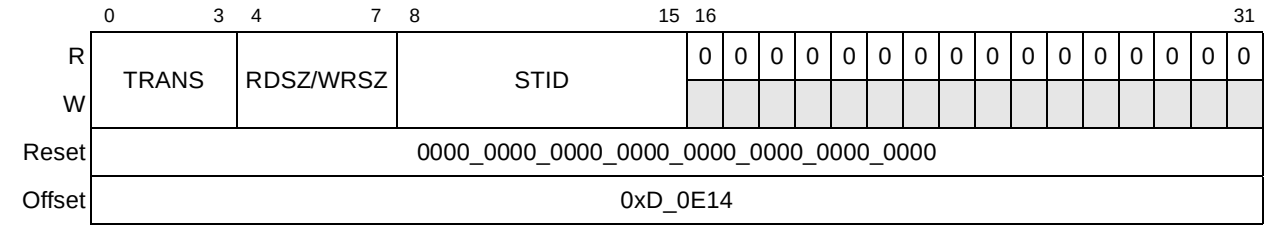


Figure 17-48. Error Packet Capture Register 1 (EPCR1)—Type 8 Request Packet Format

Table 17-50 describes EPCR1 fields for a type 8 request packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 0 or 1).

Table 17-50. 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

17.3.2.3.12 EPCR1—Type 8 Response Packet Format

Figure 17-49 describes the EPCR1 for a type 8 response packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 2 or 3).

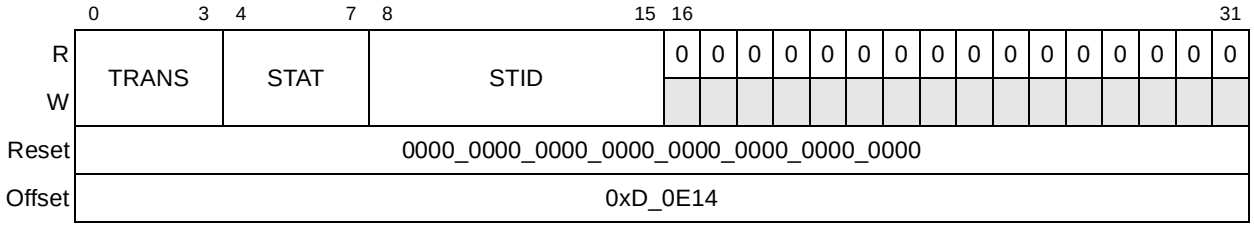


Figure 17-49. Error Packet Capture Register 1 (EPCR1)—Type 8 Response Packet Format

Table 17-51 describes the fields of the EPCR1 for a type 8 response packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 2 or 3).

Table 17-51. 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

17.3.2.3.13 EPCR1—Type 10 Packet Format

Figure 17-50 describes the EPCR1 for a type 10 packet format (EPCR0[PFT] = 10).

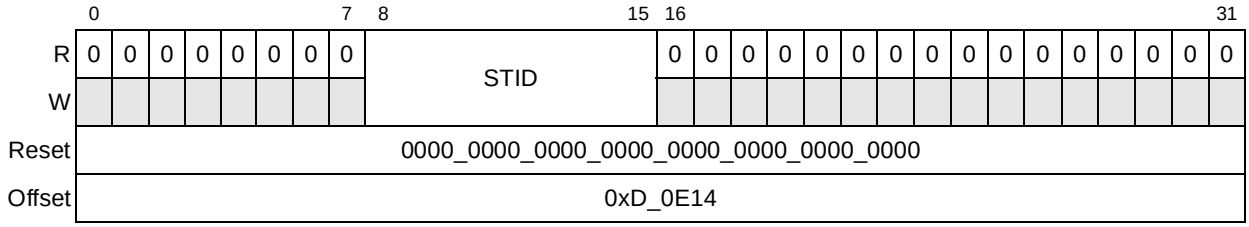


Figure 17-50. Error Packet Capture Register 1 (EPCR1)—Type 10 Packet Format

Table 17-52 describes EPCR1 fields for a type 10 packet format (EPCR0[PFT] = 10).

Table 17-52. EPCR1 Type 10 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	STID	Source transaction ID
16–31	—	Reserved

17.3.2.3.14 EPCR1—Type 11 Packet Format

Figure 17-51 describes the EPCR1 for a type 11 packet format (EPCR0[PFT] = 11).

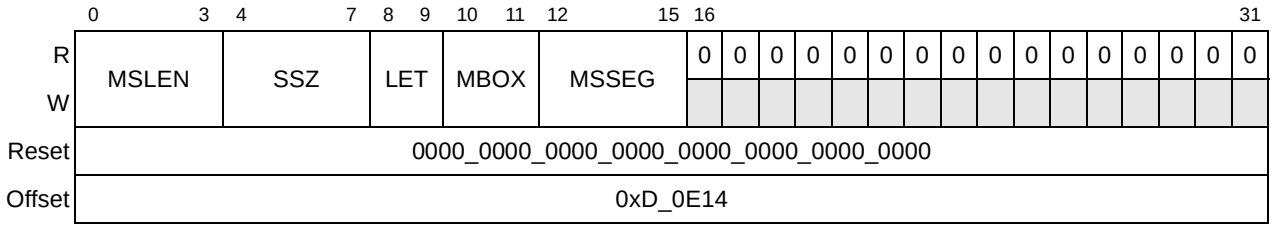


Figure 17-51. Error Packet Capture Register 1 (EPCR1)—Type 11 Packet Format

Table 17-53 describes EPCR1 fields for a type 11 packet format (EPCR0[PFT] = 11).

Table 17-53. 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

17.3.2.3.15 EPCR1—Type 13 Packet Format

Figure 17-52 describes the EPCR1 for a type 13 packet format (EPCR0[PFT] = 13).

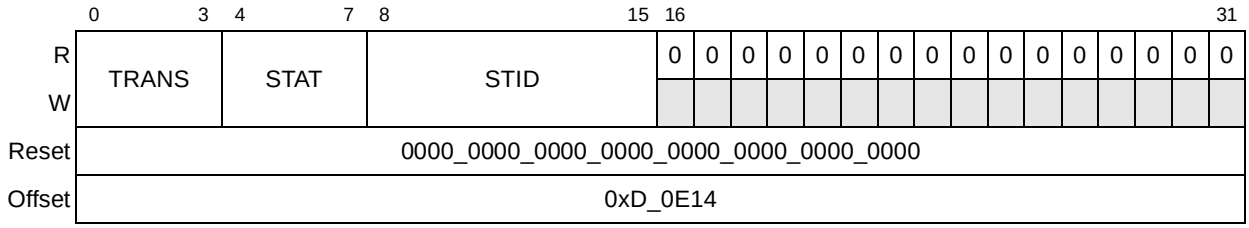


Figure 17-52. Error Packet Capture Register 1 (EPCR1)—Type 13 Packet Format

Table 17-54 describes EPCR1 fields for a type 13 packet format (EPCR0[PFT] = 13).

Table 17-54. 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

17.3.2.3.16 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 17.3.2.3.17, “EPCR2—Type 1, 2, or 5 Packet Format,” through Section 17.3.2.3.22, “EPCR2—Type 11 or 13 Packet Format.”

17.3.2.3.17 EPCR2—Type 1, 2, or 5 Packet Format

Figure 17-53 shows EPCR2 for a type 1, 2, or 5 packet format (EPCR0[PFT] = 1, 2, or 5).

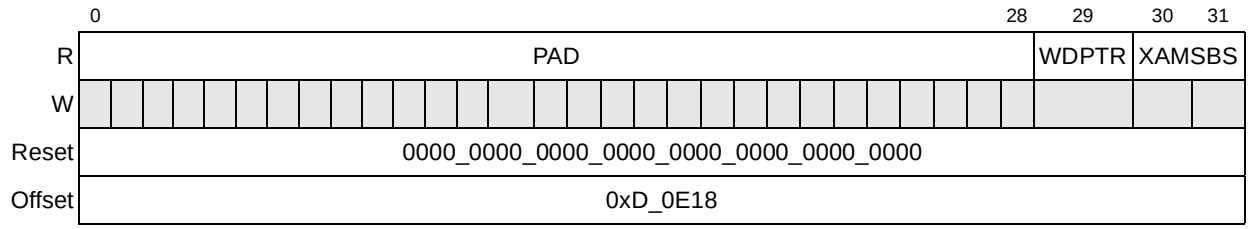


Figure 17-53. Error Packet Capture Register 2 (EPCR2)—Type 1, 2, or 5 Packet Format

Table 17-55 shows EPCR2 for a type 1, 2, or 5 packet format (EPCR0[PFT] = 1, 2, or 5).

Table 17-55. 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

17.3.2.3.18 EPCR2—Type 6 Packet Format

Figure 17-54 describes the EPCR2 for a type 6 packet format (EPCR0[PFT] = 6).

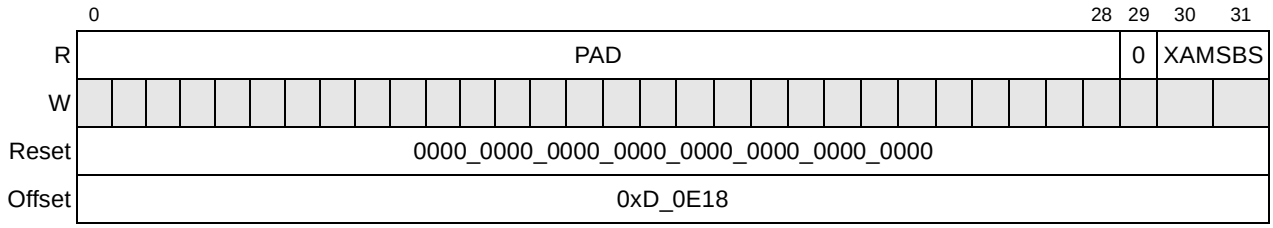


Figure 17-54. Error Packet Capture Register 2 (EPCR2)—Type 6 Packet Format

Table 17-56 describes the EPCR2 fields for a type 6 packet format (EPCR0[PFT] = 6).

Table 17-56. 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

17.3.2.3.19 EPCR2—Type 8 Request Packet Format

Figure 17-55 shows EPCR2 for a type 8 request packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 0 or 1).

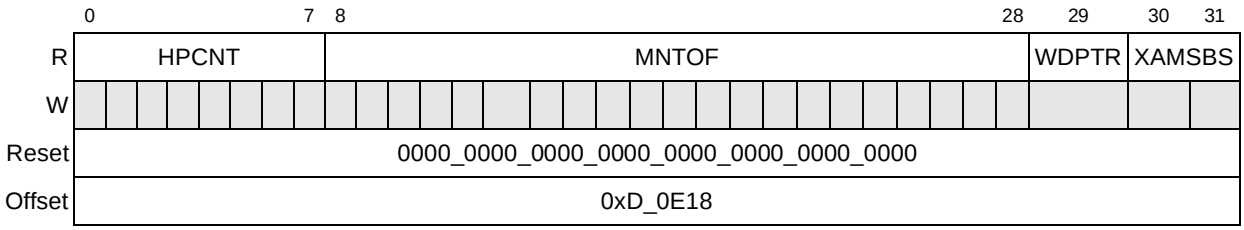


Figure 17-55. Error Packet Capture Register 2 (EPCR2)—Type 8 Request Packet Format

Table 17-57 describes the fields of the EPCR2 for a type 8 request packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 0 or 1).

Table 17-57. 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

17.3.2.3.20 EPCR2—Type 8 Response Packet Format

Figure 17-56 shows EPCR2 for a type 8 response packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 2 or 3).

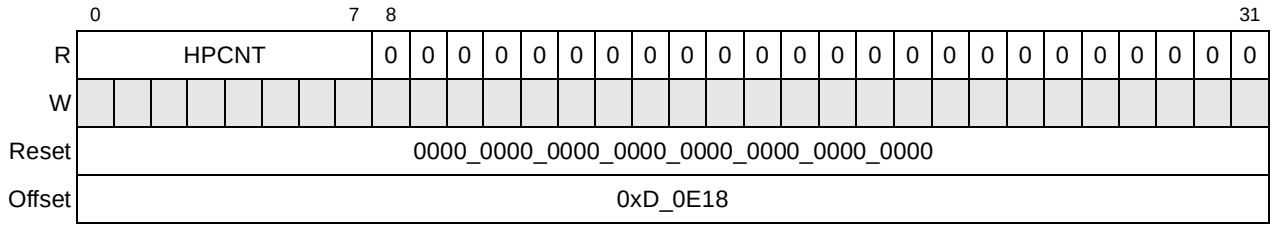


Figure 17-56. Error Packet Capture Register 2 (EPCR2)—Type 8 Response Packet Format

Table 17-58 describes EPCR2 for a type 8 response packet format (EPCR0[PFT] = 8 and EPCR1[TRANS] = 2 or 3).

Table 17-58. EPCR2 Type 8 Response Field Descriptions

Bits	Name	Description
0–7	HPCNT	Hop count field of a packet
8–31	—	Reserved

17.3.2.3.21 EPCR2—Type 10 Packet Format

Figure 17-57 describes the EPCR2 for a type 10 packet format (EPCR0[PFT] = 10).

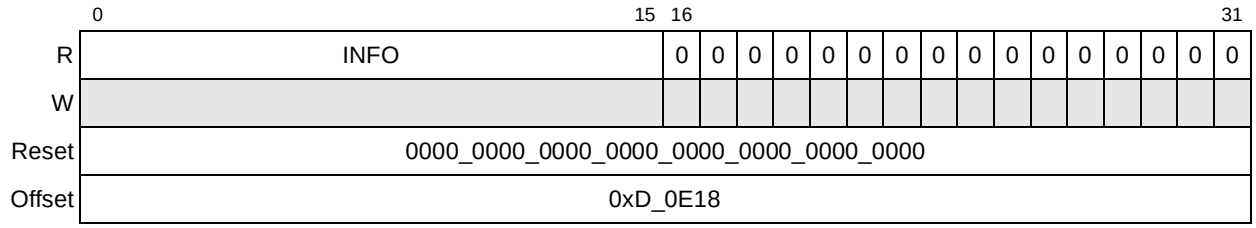


Figure 17-57. Error Packet Capture Register 2 (EPCR2)—Type 10 Packet Format

Table 17-59 describes EPCR2 for a type 10 packet format (EPCR0[PFT] = 10).

Table 17-59. EPCR2 Type 10 Field Descriptions

Bits	Name	Description
0–15	INFO	Information field of a packet
16–31	—	Reserved

17.3.2.3.22 EPCR2—Type 11 or 13 Packet Format

Figure 17-58 shows EPCR2 for a type 11 or 13 packet format (EPCR0[PFT] = 11 or 13).

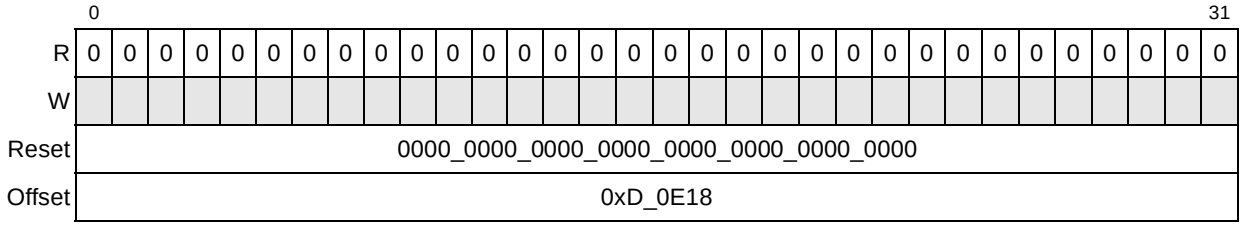


Figure 17-58. Error Packet Capture Register 2 (EPCR2)—Type 11 or 13 Packet Format

Table 17-60 describes EPCR2 for a type 11 or 13 packet format (EPCR0[PFT] = 11 or 13).

Table 17-60. EPCR2 Type 11 or 13 Field Description

Bits	Name	Description
0–31	—	Reserved

17.3.2.3.23 Port Recoverable Error Detect Register (PREDR)

The PREDR, described in Figure 17-59, indicates the recoverable errors detected on RapidIO.

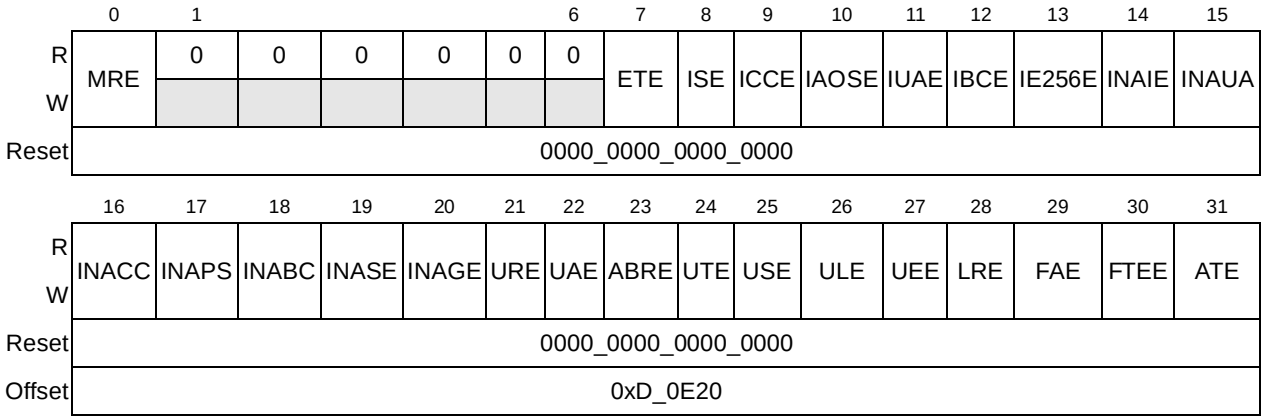


Figure 17-59. Port Recoverable Error Detect Register (PREDR)

Table 17-61 describes the fields of the PREDR.

Table 17-61. 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 17-61. 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.

17.3.2.3.24 Port Error Recovery Threshold Register (PERTR)

Figure 17-60 shows the PERTR.

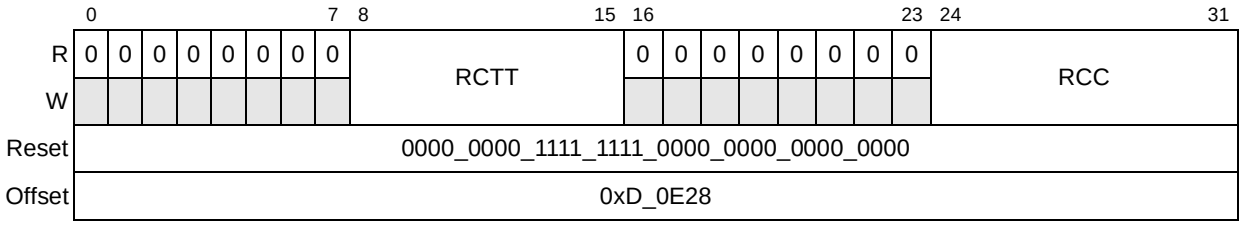


Figure 17-60. Port Error Recovery Threshold Register (PERTR)

Table 17-62 describes the fields of the PERTR.

Table 17-62. 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).

17.3.2.3.25 Port Retry Threshold Register (PRTR)

Figure 17-61 shows the PRTR.

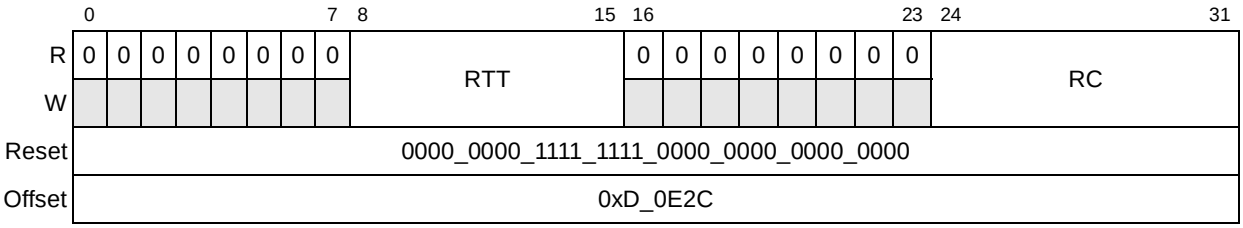


Figure 17-61. Port Retry Threshold Register (PRTR)

Table 17-63 describes the fields of the PRTR.

Table 17-63. 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).

17.3.3 RapidIO Message Unit Registers

This section describes RapidIO outbound and inbound message registers, doorbell registers and port-write registers. Refer to Section 17.8, “RapidIO Message Unit,” for additional information about how these registers affect the message unit.

17.3.3.1 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 Interconnect Specification*.

17.3.3.1.1 Outbound Mode Register (OMR)

The outbound mode register (OMR) allows software to start a message operation and to control various message operation characteristics. Figure 17-62 shows the OMR.

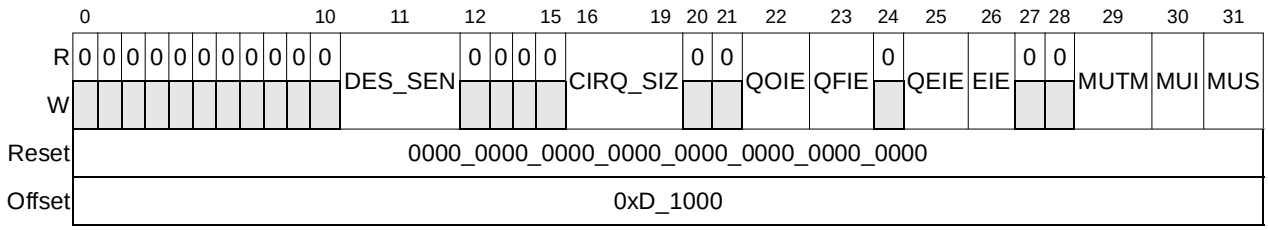


Figure 17-62. Outbound Mode Register (OMR)

Table 17-64 describes the fields of the OMR.

Table 17-64. 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 17-64. 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.

17.3.3.1.2 Outbound Status Register (OSR)

The outbound status register (OSR), shown in Figure 17-63, reports various message unit conditions during and after a message operation. Writing a 1 to a writable condition bit in OSR clears it.

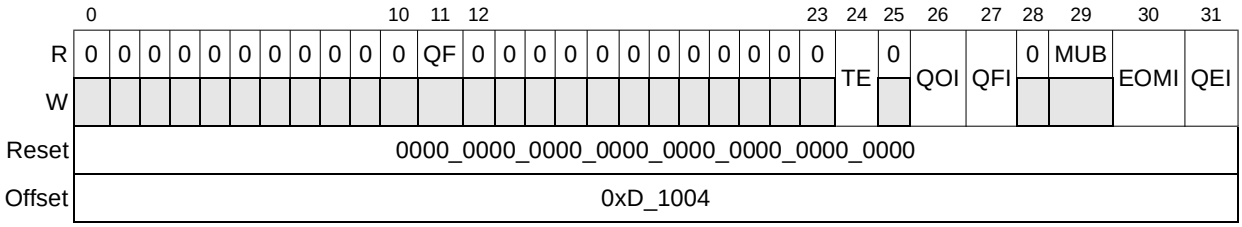


Figure 17-63. Outbound Status Register (OSR)

Table 17-65 describes the fields of the OSR.

Table 17-65. 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 17-65. 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.

17.3.3.1.3 Outbound Descriptor Queue Dequeue Pointer Address Register (ODQDPAR)

The outbound descriptor queue dequeue pointer address register (ODQDPAR), shown in Figure 17-64, 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.

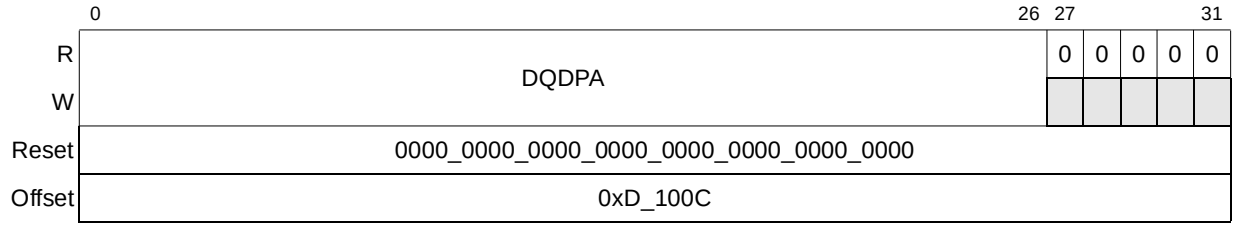


Figure 17-64. Outbound Descriptor Queue Dequeue Pointer Address Register (ODQDPAR)

Table 17-66 describes the fields of the ODQDPAR.

Table 17-66. 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

17.3.3.1.4 Outbound Unit Source Address Register (OSAR)

The outbound unit source address register (OSAR), shown in Figure 17-65, 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.

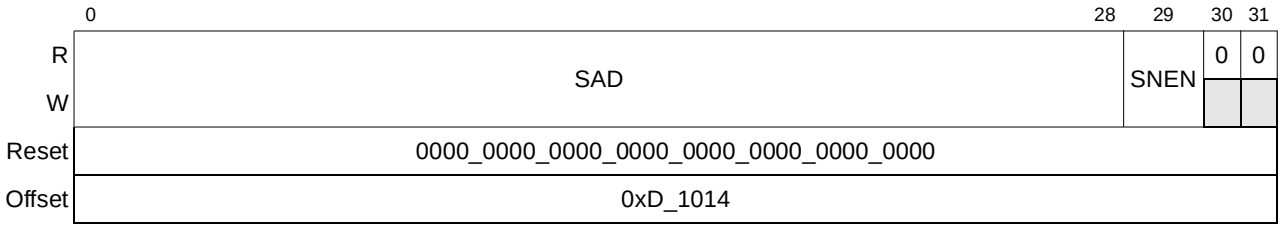


Figure 17-65. Outbound Unit Source Address Registers (OSAR)

Table 17-67 describes the fields of the OSAR.

Table 17-67. 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

17.3.3.1.5 Outbound Destination Port Register (ODPR)

The destination port register (ODPR), shown in Figure 17-66, 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.

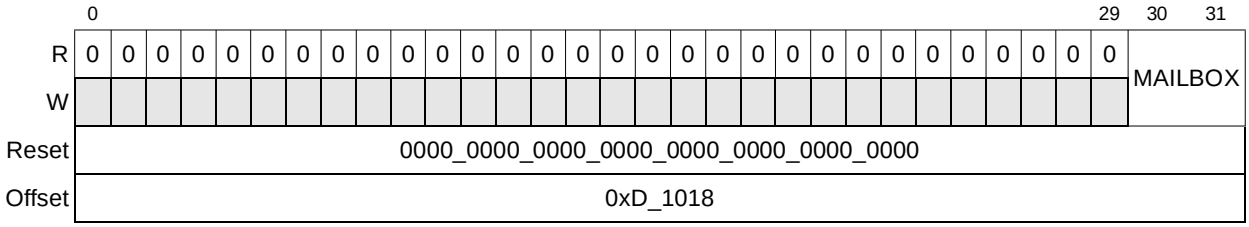


Figure 17-66. Outbound Destination Port Registers (ODPR)

Table 17-68 describes the fields of the ODPR.

Table 17-68. ODPR Field Descriptions

Bits	Name	Description
0–29	—	Reserved
30–31	MAILBOX	Value for mbox field in message packet

17.3.3.1.6 Outbound Destination Attributes Register (ODATR)

The outbound destination attributes register (ODATR) contains the transaction attributes to be used for the message operation. Figure 17-67 shows the ODATR.

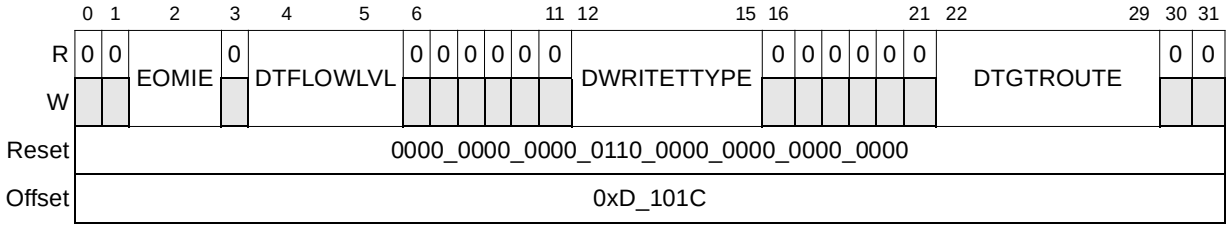


Figure 17-67. Outbound Destination Attributes Register (ODATR)

Table 17-69 describes the fields of the ODATR.

Table 17-69. 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 17-69. 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

17.3.3.1.7 Outbound Double-Word Count Register (ODCR)

The outbound double-word count register (ODCR), shown in Figure 17-68, contains the number of double words for the message operation. The maximum message operation size is 4 Kbytes and the minimum is 8 bytes.

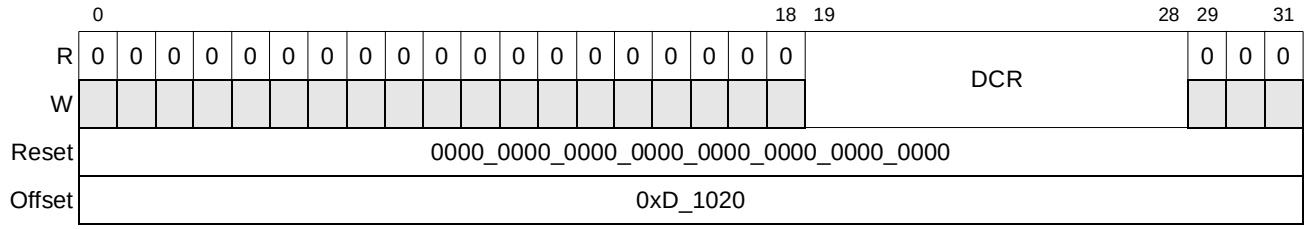


Figure 17-68. Outbound Double-Word Count Register (ODCR)

Table 17-70 describes the fields of the ODCR.

Table 17-70. 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 0000Reserved 00 0000 00018 bytes 00 0000 001016 bytes 00 0000 010032 bytes 00 0000 100064 bytes 00 0001 0000128 bytes 00 0010 0000256 bytes 00 0100 0000512 bytes 00 1000 00001024 bytes 01 0000 00002048 bytes 10 0000 00004096 bytes All other values yield undefined behavior.
29–31	—	Reserved

17.3.3.1.8 Outbound Descriptor Queue Enqueue Pointer Address Register (ODQEPAR)

The outbound descriptor queue enqueue pointer address register (ODQEPAR), shown in Figure 17-69, 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).

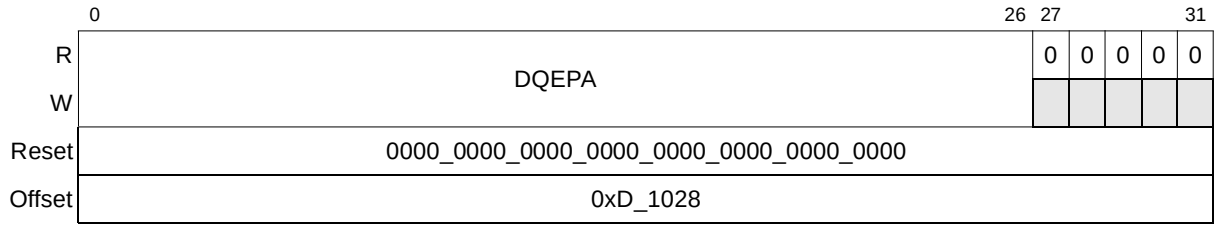


Figure 17-69. Outbound Descriptor Queue Enqueue Pointer Register (ODQEPAR)

Table 17-71 describes the fields of the ODQEPAR.

Table 17-71. 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

17.3.3.2 RapidIO Inbound Message Registers

Registers in this section describe the RapidIO inbound message registers.

17.3.3.2.1 Inbound Mode Register (IMR)

The inbound mode register (IMR) allows software to enable the mailbox controller and to control various message operation characteristics. Figure 17-70 shows the IMR.

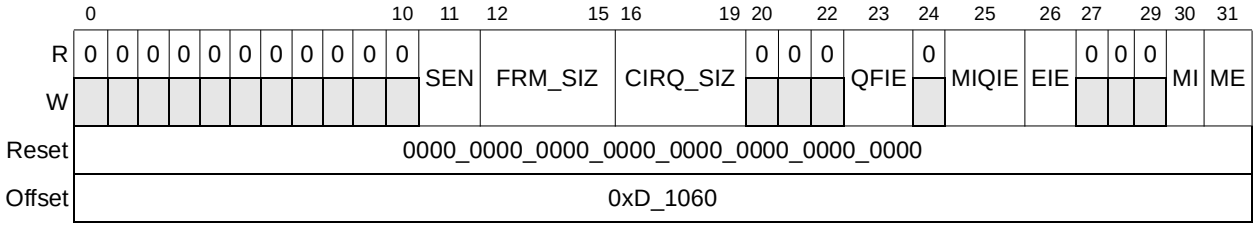


Figure 17-70. Inbound Mode Register (IMR)

Table 17-72 describes the fields of the IMR.

Table 17-72. 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 17-72. 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.

17.3.3.2.2 Inbound Status Register (ISR)

The inbound status register (ISR), shown in Figure 17-71, reports mailbox conditions during and after a message operation. Writing a 1 to a writable condition bit in ISR clears it.

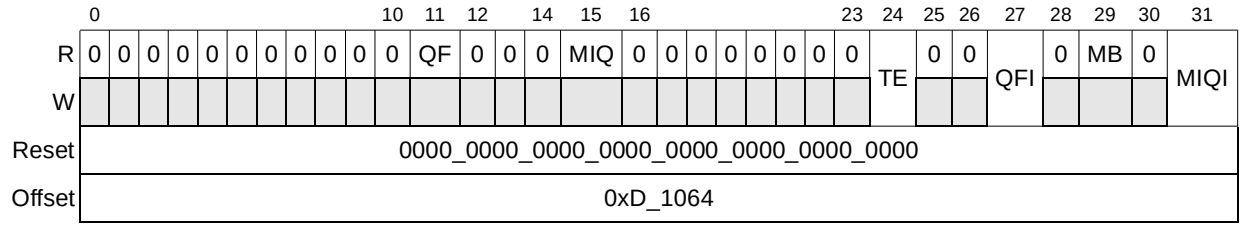


Figure 17-71. Inbound Status Register (ISR)

Table 17-73 describes the fields of the ISR.

Table 17-73. 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 17-73. 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.

17.3.3.2.3 Inbound Frame Queue Dequeue Pointer Address Register (IFQDPAR)

The inbound frame queue dequeue pointer address register (IFQDPAR), shown in Figure 17-72, 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.

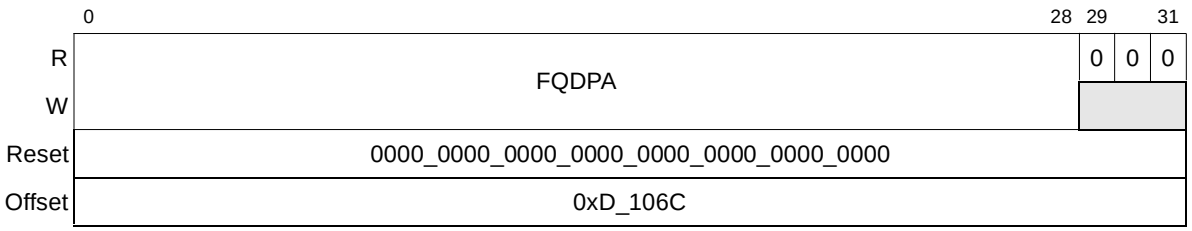


Figure 17-72. Inbound Frame Queue Dequeue Pointer Address Registers (IFQDPAR)

Table 17-74 describes the fields of the IFQDPAR.

Table 17-74. 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

17.3.3.2.4 Inbound Frame Queue Enqueue Pointer Address Register (IFQEPAR)

The inbound frame queue enqueue pointer address register (IFQEPAR), shown in Figure 17-73, 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.

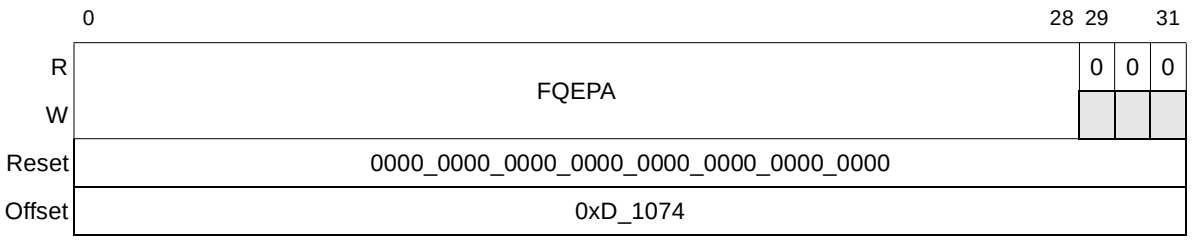


Figure 17-73. Inbound Frame Queue Enqueue Pointer Address Registers (IFQEPAR)

Table 17-75 describes the fields of the IFQEPAR.

Table 17-75. 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

17.3.3.3 RapidIO Doorbell Registers

These registers control the RapidIO doorbell unit. The following sections provide descriptions of these registers.

17.3.3.3.1 Doorbell Mode Register (DMR)

The doorbell mode register (DMR) allows software to enable the doorbell controller to control various doorbell operation characteristics. Figure 17-74 shows the DMR.

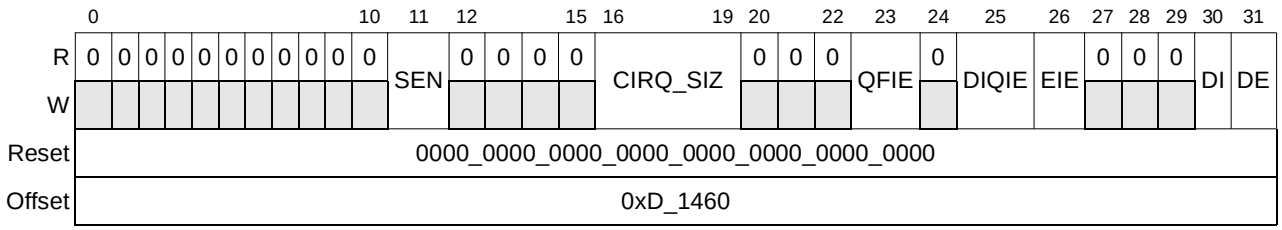


Figure 17-74. Doorbell Mode Register (DMR)

Table 17-76 describes the fields of the DMR.

Table 17-76. 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 17-76. 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.

17.3.3.3.2 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 17-75 shows the DSR.

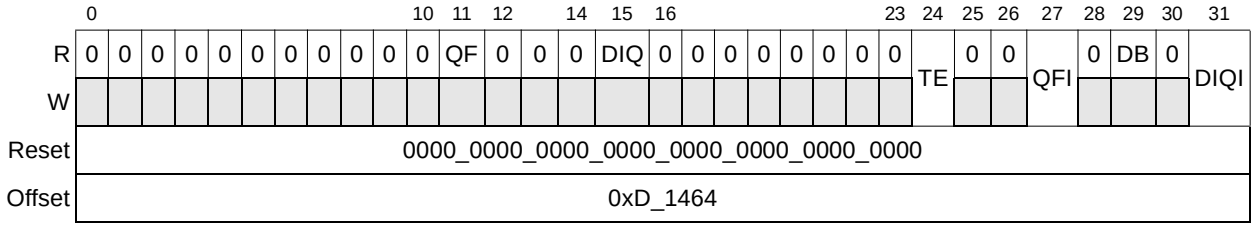


Figure 17-75. Doorbell Status Register (DSR)

Table 17-77 describes the fields of the DSR.

Table 17-77. 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.

Table 17-77. DSR Field Descriptions (continued)

Bits	Name	Description
12–14	—	Reserved
15	DIQ	Doorbell-in-queue (read only) 0 No doorbell-in-queue conditions 1 If the queue goes from empty to not empty, 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)

17.3.3.3.3 Doorbell Queue/Dequeue Pointer Address Register (DQDPAR)

The doorbell queue dequeue pointer address register (DQDPAR), shown in Figure 17-76, 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.

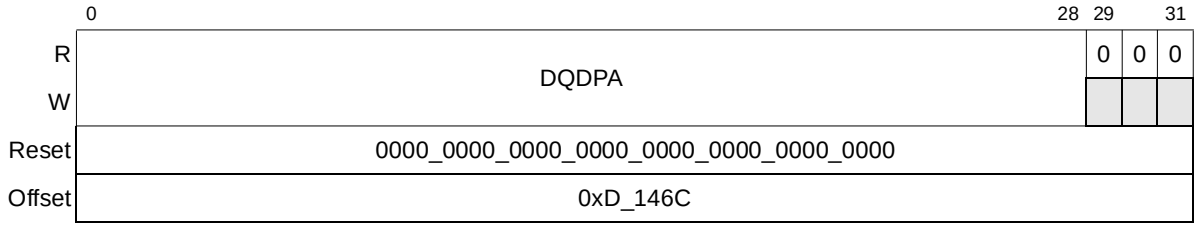


Figure 17-76. Doorbell Queue Dequeue Pointer Address Registers (DQDPAR)

Table 17-78 describes the fields of the DQDPAR.

Table 17-78. 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

17.3.3.3.4 Doorbell Queue Enqueue Pointer Address Register (DQEPAR)

The doorbell queue enqueue pointer address register (DQEPAR), shown in Figure 17-77, 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.

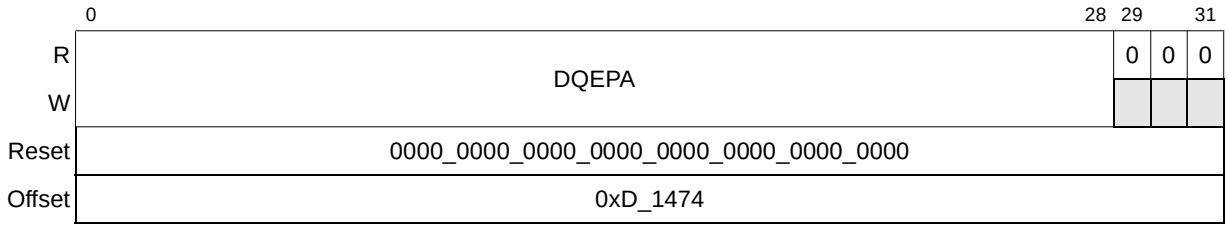


Figure 17-77. Doorbell Queue Enqueue Pointer Address Register (DQEPAR)

Table 17-79 describes the fields of the DQEPAR.

Table 17-79. 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

17.3.3.4 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 Interconnect Specification*.

17.3.3.4.1 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 17-78 shows the PWMR.

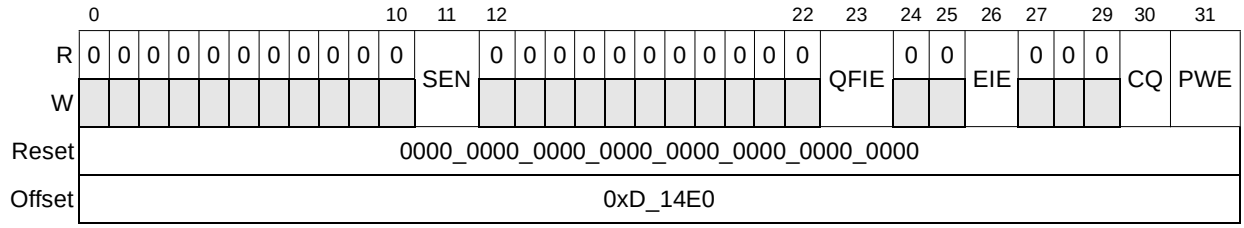


Figure 17-78. Port-Write Mode Register (PWMR)

Table 17-80 describes the fields of the PWMR.

Table 17-80. 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 17-80. 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.

17.3.3.4.2 Port-Write Status Register (PWSR)

The port-write status register (PWSR), shown in Figure 17-79, reports various port-write conditions after a port-write operation. Writing a 1 to a corresponding set bit clears the bit.

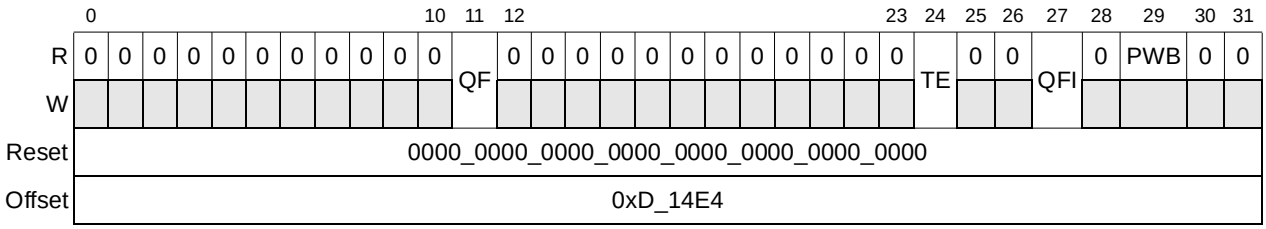


Figure 17-79. Port-Write Status Register (PWSR)

Table 17-81 describes the fields of the PWSR.

Table 17-81. 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 17-81. 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

17.3.3.4.3 Port-Write Queue Base Address Register (PWQBAR)

The port-write queue base address register (PWQBAR), shown in Figure 17-80, contains the 64-byte cache line address for the port-write data payload. PWQBAR must be initialized by software.

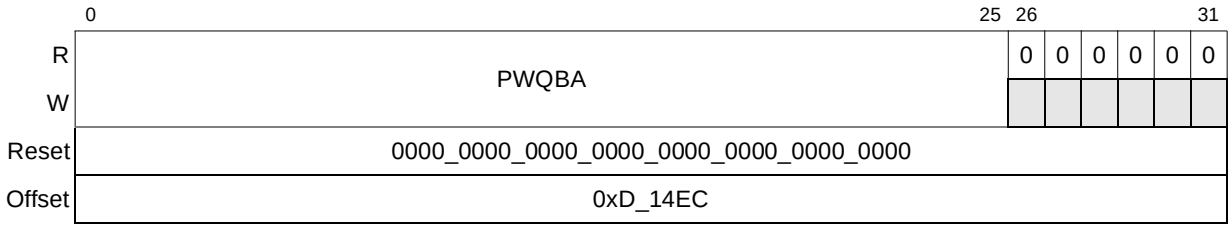


Figure 17-80. Port-Write Queue Base Address Register (PWQBAR)

Table 17-82 describes the fields of the PWQBAR.

Table 17-82. 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

17.4 Functional Description

The following sections provide summaries of RapidIO transactions, packets and control symbols. Additional information may be found in the *Parallel RapidIO Interconnect Specification*.

17.4.1 RapidIO Transaction, Packet, and Control Symbol Summary

Table 17-83 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 17-83.

Table 17-83. 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 17-83. 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 17-84 summarizes packet formats. The following definitions and Figure 17-81 accompany the table.

- PMF—Physical maintenance field; prefixed to each packet format (see Figure 17-81)
- 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 17-81. Physical Maintenance Field (PMF) Definition

Table 17-84. 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>
</table>

Table 17-84. RapidIO Packet Format Summary (TT = 0b00) (continued)

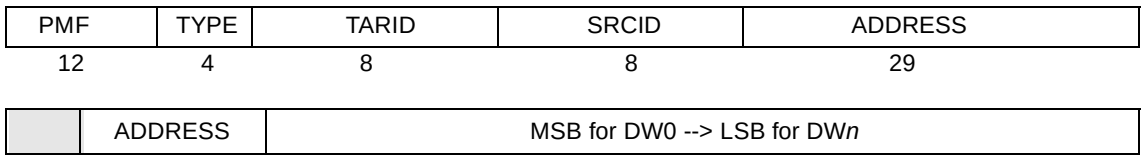
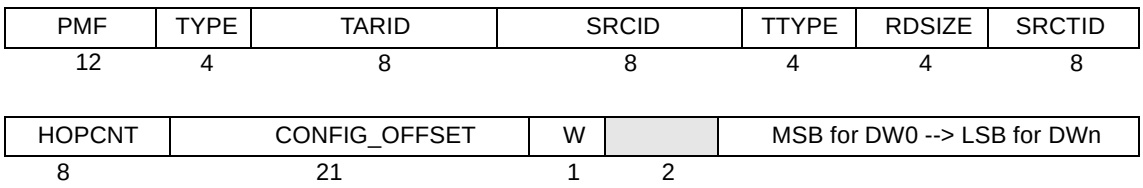
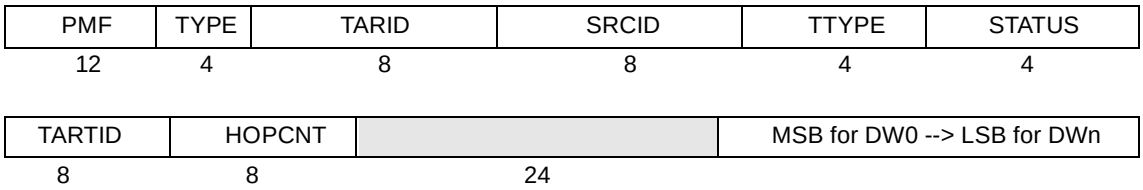
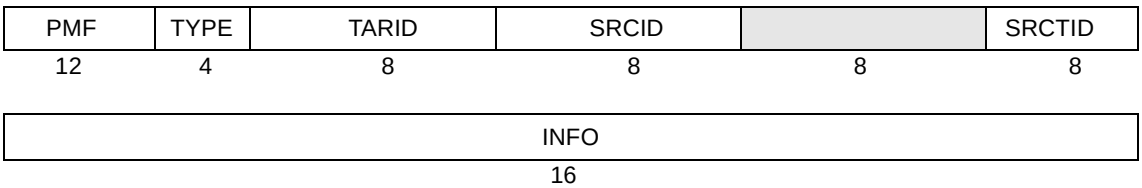
Type	Packet Format
6 (0110)	Streaming-Write Class: 
8 (1000)	Maintenance Class Request:  TTYPE: 0000 (read); 0001 (write): 0100 (port-write) Maintenance Class Response:  TTYPE: 0010 (read response); 0011 (write response) STATUS: 0000 (done); 0111 (error);
10 (1010)	Doorbell Class: 

Table 17-84. RapidIO Packet Format Summary (TT = 0b00) (continued)

Type	Packet Format
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 17-85 summarizes the defined control symbols. Implementation details for this particular RapidIO controller are noted in the comments section of the table.

Table 17-85. 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
2 (010)	Packet not accepted: S = 1 (1) | TgtAckID (3) | 00000 (5) | 1 (1) | CAUSE (3) | 010 (3)	Cause 000 Encountered internal error 001 Received unexpected ackID on packet 010 Uncorrectable error on control symbol 011 Input port has been stopped 100 Received bad CRC on packet 111 General error

Table 17-85. Control Symbol Formats (continued)

Symbol Type	Symbol Format	Comments
4 (100)	Packet control: S = 1 (1) | SUB_TYPE (3) | 00000 (5) | CONTENTS (4) | 100 (3)	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: S = 1 (1) | CMD (3) | 00000 (5) | BUF_STATUS (4) | 100 (3)	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: S = 1 (1) | ACK_STAT (3) | 00000 (5) | LINK_STAT (4) | 100 (3)	Ack status 000 Expecting ackID 0 001 Expecting ackID 1 .... 111 Expecting ackID 7 Link status 0000 Un-initialized 0001 Initializing 0010 Error 0011 Stopped 0100 Retry-stopped 0101 Error-stopped 1000 Valid, ackID 0 .... 111 Valid, ackID 7

17.5 RapidIO Functionality

This section describes RapidIO general device functionality, common transport layer functionality, and logical layer functionality.

17.5.1 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 17-86 lists general requirements for a RapidIO-compliant device.

Table 17-86. 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
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

17.5.1.1 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 17-87 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 17-87. 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
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

Table 17-87. General Device 8/16 LP-LVDS Physical Layer Basic Functionality List (continued)

Item No.	Item Functionality	Meets Reqs?
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 17-88 lists requirements for link maintenance functionality and the compliance of the RapidIO controller.

Table 17-88. 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 17-89 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 17-89. 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 17-89. 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 17-90 lists the requirements for the device to receive a packet.

Table 17-90. 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 17-90. 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 17-91 lists recoverable errors at the physical layer.

Table 17-91. 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 17-91. 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 17-91. 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 17-92 lists nonrecoverable errors at the physical layer.

Table 17-92. 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

17.5.2 Common Transport Layer Functionality Lists

The common transport layer functionality is subdivided into individual lists.

Table 17-93 lists the basic functionality for the device at the common transport layer and the compliance of the RapidIO controller.

Table 17-93. 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 17-94 lists the functionality for this controller in transmitting a packet (a slave device is required to generate response packets).

Table 17-94. 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 17-95 lists the functionality for this controller in receiving a packet.

Table 17-95. 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 17-96 lists detectable errors.

Table 17-96. General Device Common Transport Layer Detectable Errors List

Item No.	Item Functionality	Meets Reqs?
1	Received reserved TT field encoding	Y

17.5.3 Logical Layer Functionality Lists

The logical layer functionality is subdivided into individual lists.

Table 17-97 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 17-97. 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 17-98 lists requirements for this controller when functioning as a transaction target.

Table 17-98. 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

17.5.3.1 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 17-99 lists the errors detected by the RapidIO controller.

Table 17-99. 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

17.5.3.2 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 Interconnect Specification* except for atomic test and swap and atomic swap transactions.

17.6 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 Interconnect 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 17-100 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 17-100 lists the recoverable errors detected by the RapidIO implementation and the resulting actions taken.

Table 17-100. 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 17-100. 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 Interconnect 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 17-100. 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 17-101 lists the notification errors detected by the RapidIO implementation and the resulting actions taken.

Table 17-101. 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 17-102 lists the fatal errors detected by the RapidIO implementation and the resulting actions taken.

Table 17-102. 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)
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)

Table 17-102. Fatal Errors Detected by RapidIO Controller (continued)

Error	Description	Action	Capture Registers
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)

17.7 ATMU (Address Translation and Mapping Unit)

Outbound address translation refers to the translation of an address from the local address space to that of RapidIO. In the same context, inbound address translation and mapping refers to the translation of an address from the external address space of RapidIO to the local address space understood by the internal interfaces.

Although, the *Parallel RapidIO Interconnect 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.

17.7.1 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.

17.7.1.1 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.

17.7.1.2 Outbound Special Transactions and Requirements

Due to requirements of the *Parallel RapidIO Interconnect 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 Interconnect 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 Interconnect Specification*.

The *Parallel RapidIO Interconnect Specification* should be referenced for any additional requirements that are not listed in this section.

17.7.2 Inbound ATMU Translation

Inbound ATMU windows perform the address translation from the 34-bit external RapidIO address space to the 32-bit internal address space. Inbound ATMU windows also attach attributes,

transaction type, and the target interface to the transaction. The RapidIO controller has four inbound ATMU windows plus the default window 0. The default window is always enabled. Overlapping inbound window hits result in the use of the lowest number window register set hit.

17.7.2.1 Inbound ATMU LCSBA1CSR Window

There is an additional inbound window that is used for inbound translation. It is a special condition for external accesses to LCSBA1CSR (local configuration space base address register) address space. The purpose of the LCSBA1CSR window is to allow external devices to access the LCSBA1CSR address space without knowledge of the internal memory map. When a remote device attempts to access the LCSBA1CSR register set (address hitting LCSBA1CSR window), the lower 20 bits are used as the offset to access the specific register in the LCSBA1CSR address space. The LCSBA1CSR window has the highest priority for inbound translation. If the inbound transaction's address does not hit the LCSBA1CSR window or any of the four inbound ATMU windows (1–4), the translation attributes defined by the default window (window 0) are used.

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.

17.7.2.2 Inbound ATMU Bypass Mode

Just as some outbound transactions may bypass outbound ATMU translation, some inbound transactions may also bypass ATMU translation because the address associated with the original transaction is already mapped to the appropriate address space. Inbound messages, doorbells and port-writes go through a different form of address mapping and are not mapped through the ATMU. These transactions have the stash and cache_lock attributes 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 17.8.4.1, "Data Message Controller," Section 17.8.4.4, "Doorbell Message Controller," and Section 17.8.4.5, "Port-Write Controller Structure."

17.7.2.3 Inbound Special Transactions and Requirements

Just as with outbound translation, some inbound transactions require special handling to comply with the *Parallel RapidIO Interconnect 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 Interconnect 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 Interconnect Specification* should be referenced for any additional requirements that are not listed in this section.

17.7.2.4 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 17.3.2.3, "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.

17.8 RapidIO Message Unit

This description is an extension of Section 17.7.1, "Outbound ATMU Translation," and Section 17.7.2, "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 Interconnect Specification*.

17.8.1 Overview

The RapidIO message unit supports a message passing programming model for inter-processor and inter-device communication. This model enables a producer to send a message across the interconnect fabric to a consumer's message hardware, called a mailbox. The receiving mailbox 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 17.3.3.1, "RapidIO Outbound Message Registers," Section 17.3.3.2, "RapidIO Inbound Message Registers," and Section 17.3.3.3, "RapidIO Doorbell Registers."

17.8.2 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

17.8.3 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.

17.8.4 RapidIO Messaging Description

This section describes controller operations, formats, interrupts, and response conditions.

17.8.4.1 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.

17.8.4.2 Outbox Controller Operation

The outbox controller is responsible for sending messages stored in a circular queue in local memory. The outbox controller supports two modes of operation: direct and chaining mode. In

direct mode, software programs the necessary registers to point to the beginning of the message in memory. In chaining mode, software programs the necessary registers to point to the beginning of the first valid descriptor in memory. The hardware then reads the descriptor to load all the necessary registers to start the message transfer.

17.8.4.2.1 Direct Mode Operation

In direct mode, OMR[MUTM] is set 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.

17.8.4.2.2 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 17-82 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.

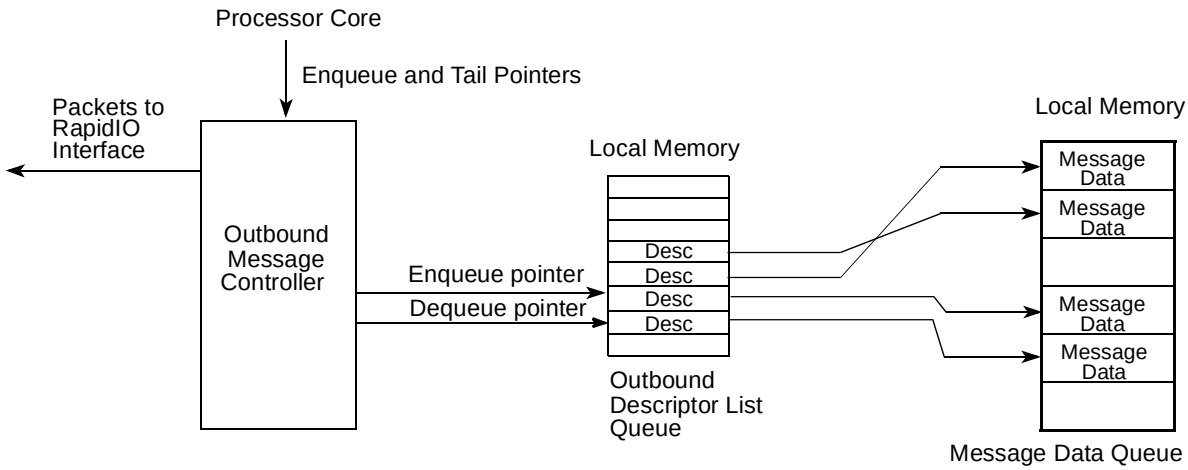


Figure 17-82. Outbound Message Queue Structure

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.

17.8.4.2.3 Switching Between Direct and Chaining Modes

The message unit architecture allows switching from direct mode to chaining mode and vice-versa 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.

17.8.4.2.4 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 17-103 describes the outbound message unit descriptor.

Table 17-103. 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 17-83 shows the queue dequeue pointer and an associated descriptor. The descriptor is only valid if the enqueue and dequeue pointers are not equal.

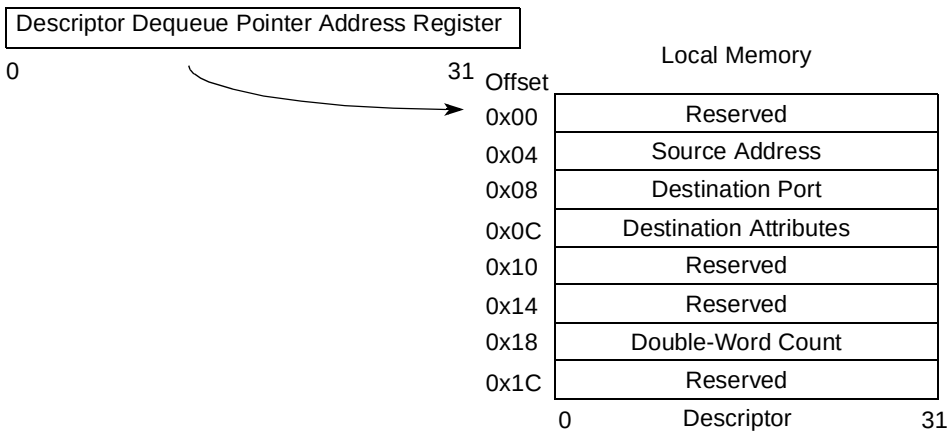


Figure 17-83. Descriptor Dequeue Pointer and Descriptor

17.8.4.2.5 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.

17.8.4.2.6 Special Error Case Condition

If a data message is in progress and the outbox encounters an error with one of the segments, it causes an error at the recipient by placing an illegal field in the packet to cause the error. This 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.

17.8.4.3 Inbox Controller Operation

The inbox controller is responsible for receiving messages and placing them in a circular queue in local memory. Although, it only supports one mailbox and one letter, 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 17-84 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.

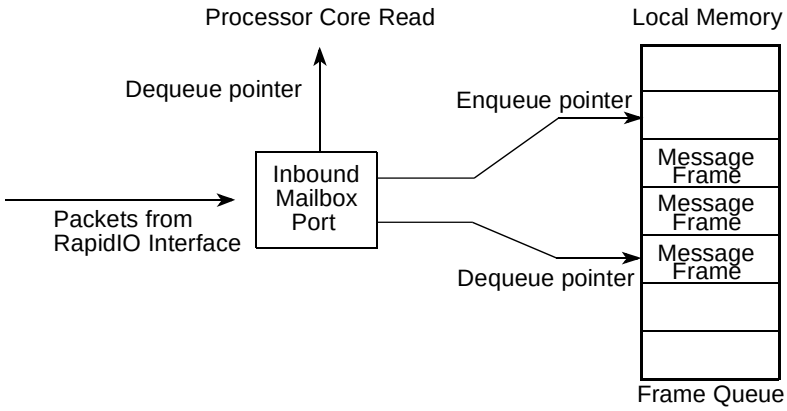


Figure 17-84. Inbound Mailbox Structure

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.

17.8.4.3.1 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

17.8.4.3.2 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

17.8.4.3.3 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.

17.8.4.3.4 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.

17.8.4.4 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 17-85 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.

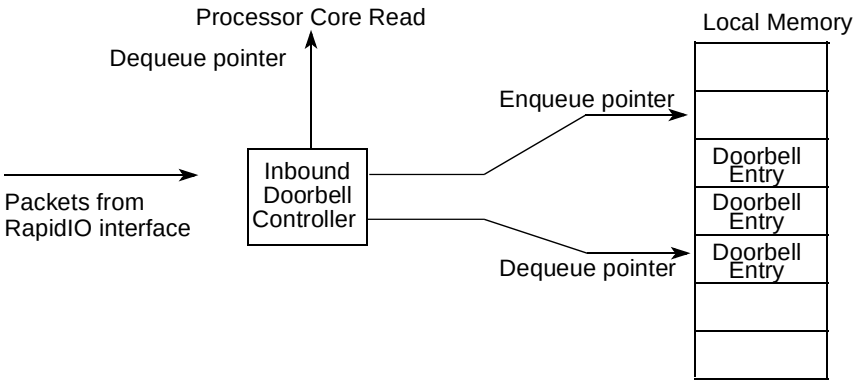


Figure 17-85. Inbound Doorbell Queue and Pointer Structure

17.8.4.4.1 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.

17.8.4.4.2 Doorbell Queue Entry Format

This section defines the format of the doorbell information written to memory by the doorbell controller. Each doorbell entry in the queue has two 32-bit offsets, one for target information and one for source information. Table 17-104 shows the target information.

Table 17-104. Target Information Definition

Bits	Name	Description
0–23	—	Reserved
24–31	TID	Target ID field from the received doorbell packet

Table 17-105 shows the source information.

Table 17-105. 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 17-86 shows the doorbell queue entry fields and their related offsets.

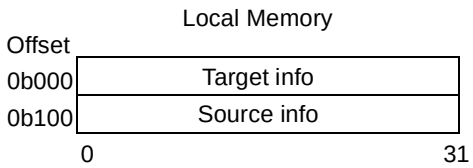


Figure 17-86. Doorbell Entry Format

17.8.4.4.3 Retry Response Conditions

There is one condition in which a doorbell message is logical layer retried (response retry):

- Doorbell request received and the doorbell circular queue is full

17.8.4.4.4 Error Response Conditions

Several conditions cause a logical layer error response (response error):

- Doorbell controller is error state
- Doorbell controller is not enabled

17.8.4.4.5 Doorbell Controller Interrupts

The doorbell controller generates 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

17.8.4.5 Port-Write Controller Structure

The implementation of the port-write controller is very similar to the inbound message hardware. The port-write is intended as an error reporting mechanism from an end-point-free device to a control processor or other system host.

Figure 17-87 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.

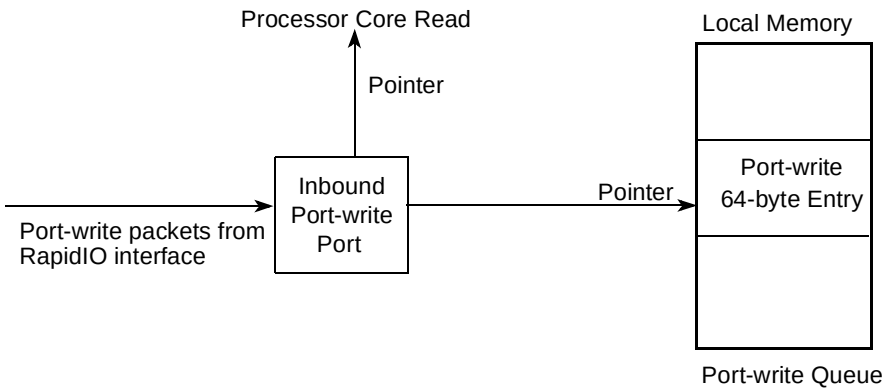


Figure 17-87. Inbound Port-Write Structure

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.

17.9 Initialization and Application Information

The following steps outline the RapidIO port initialization procedure:

1. Configure the transmit clock select pins to select either the receive clock, externally provided transmit clock or the platform clock for the RapidIO transmit clock domain.
2. Initialize the host/agent field (using configuration pins) to put the end-point in host or agent mode over RapidIO.
3. Initialize the Device ID field (using configuration pins) to assign a device identifier at power-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 Interconnect Specification* for configuring the RapidIO system.

Part IV
Global Functions and Debug

Part IV defines other global blocks of the MPC8540. The following chapters are included:

- Chapter 18, “Global Utilities,” defines the global utilities of the MPC8540. These include power management, I/O device enabling, power-on reset (POR) configuration monitoring, general-purpose I/O signal use, and multiplexing for the interrupt and local bus chip select signals
- Chapter 19, “Performance Monitor,” describes the performance monitor of the MPC8540.
- Chapter 20, “Debug Features and Watchpoint Facility,” describes the debug features and watchpoint monitor of the MPC8540.
- Chapter 21, “10/100 Fast Ethernet Controller,” includes information about using the dual-speed Ethernet controller of the MPC8540 for debug purposes.

Chapter 18
Global Utilities

This chapter describes the global utilities of the MPC8540. It provides signal descriptions, register descriptions, and a functional description of these utilities.

18.1 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.

18.2 Global Utilities Features

This section provides an overview of global utilities features.

18.2.1 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)

18.2.2 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.

18.2.3 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 $\overline{\text{HRESET}}$.

18.2.4 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.

18.2.5 Clock Control

The global utilities block also selects the internal clock signal driven on CLK_OUT.

18.3 External Signal Descriptions

The following subsections provide information about signals that serve as global utilities.

18.3.1 Signals Overview

Table 18-1 summarizes the external signals used by the global utilities block.

Table 18-1. External Signal Summary

Signal Name	I/O	Description	Reference (Section/Page)
ASLEEP	O	Signals that the device has reached a sleep state	18.5.1.5.3/18-24
$\overline{\text{CKSTP\_ IN}}$	I	Checkstop input	—
$\overline{\text{CKSTP\_OUT}}$	O	Checkstop output	—
CLK_OUT	O	Clock out. Selected by CLKOCR values	18.4.1.16/18-18

18.3.2 Detailed Signal Descriptions

Table 18-2 describes signals in the global utilities block in detail.

Table 18-2. Detailed Signal Descriptions

Signal	I/O	Description	
ASLEEP	O	Asleep. See Section 18.5.1.5.3, “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 18-2. 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 MPC8540 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 MPC8540 is reset with assertion of HRESET
CKSTP_OUT	O	Checkstop out	
		State Meaning	Asserted—Indicates that the e500 core of the MPC8540 is in a checkstop state. The rest of the MPC8540 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 18.4.1.16, “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]

18.4 Memory Map/Register Definition

Table 18-3 summarizes the global utilities registers and their addresses.

Table 18-3. Global Utilities Block Register Summary

Offset	Register	Access	Reset	Section/Page
Power-On Reset Configuration Values				
0xE_0000	PORPLLSR—POR PLL ratio status register	R	0x00*nn*_00*nn*	18.4.1.1/18-4
0xE_0004	PORBMSR—POR boot mode status register	R	0x*nnnn*_0000	18.4.1.2/18-5
0xE_0008	PORIMPSCR—POR I/O impedance status and control register	R/W	0x000*n*_007F	18.4.1.3/18-6
0xE_000C	PORDEVSR—POR I/O device status register	R	See ref.	18.4.1.4/18-7
0xE_0010	PORDBGMSR—POR debug mode status register	R	See ref.	18.4.1.5/18-9
0xE_0020	GPPORCR—General-purpose POR configuration register	R	See ref.	18.4.1.6/18-9
Signal Multiplexing and GPIO Controls				
0xE_0030	GPIOCR—GPIO control register	R/W	0x0000_0000	18.4.1.7/18-10
0xE_0040	GPOUTDR—General-purpose output data register	R/W	0x0000_0000	18.4.1.8/18-11

Table 18-3. Global Utilities Block Register Summary (continued)

Offset	Register	Access	Reset	Section/Page
0xE_0050	GPINDR—General-purpose input data register	R	0x*nnnn*_0000	18.4.1.9/18-12
0xE_0060	PMUXCR—Alternate function signal multiplex control	R/W	0x0000_0000	18.4.1.9/18-12
Device Disables				
0xE_0070	DEVDISR—Device disable control	R/W	0x0000_0000	18.4.1.11/18-13
Power Management Registers				
0xE_0080	POWMGTCSR—Power management status and control register	R/W	0x0000_0000	18.4.1.12/18-15
Interrupt Reporting				
0xE_0090	MCPSUMR—Machine check summary register	Read/Clear	0x0000_0000	18.4.1.13/18-16
Version Registers				
0xE_00A0	PVR—Processor version register	R	e500 processor version	18.4.1.14/18-17
0xE_00A4	SVR—System version register	R	MPC8540 system version	18.4.1.15/18-18
Debug Control				
0xE_0E00	CLKOCR—Clock out select register	R/W	0x0000_0000	18.4.1.16/18-18
0xE_0E10	DDRDLLCR—DDR DLL control register	R/W	0x0000_0000	18.4.1.17/18-19
0xE_0E20	LBDLLCR—LBC DLL control register	R/W	0x0000_0000	18.4.1.18/18-20

18.4.1 Register Descriptions

This section describes the global utilities registers in detail.

18.4.1.1 POR PLL Status Register (PORPLLSR)

PORPLLSR, shown in Figure 18-1, 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.

	0								9	10		15	16								25	26			30	31
R	0	0	0	0	0	0	0	0	0	e500_Ratio			0	0	0	0	0	0	0	0	0	Plat_Ratio			0
W																									
Reset	0000_0000_00*nn*_*nnnn*_0000_0000_00*nn*_*nnnn*																								
Offset	0xE_0000																								

Figure 18-1. POR PLL Status Register (PORPLLSR)

Table 18-4 describes the bit settings of PORPLLSR.

Table 18-4. PORPLLSR Field Descriptions

Bits	Name	Description
0–9	—	Reserved
10–15	e500_Ratio	Clock ratio between the e500 core and the CCB clock. 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. 0001002:1 0001015:2 0001103:1 0001117:2
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. 000102:1 010008:1 000113:1 010019:1 001004:1 0101010:1 001015:1 0110012:1 001106:1 1000016:1
31	—	Reserved

18.4.1.2 POR Boot Mode Status Register (PORBMSR)

The PORBMSR, shown in Figure 18-2, 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	0	0	0	0
W																																			
Reset	*n*000_0*nn*n_00*nn*_00*nn*_0000_0000_0000_0000																																		
Offset	0xE_0004																																		

Figure 18-2. POR Boot Mode Status Register (PORBMSR)

For information about how the RapidIO interface is configured, see Section 17.3.1.19, “Port General Control Command and Status Register (PGCCSR).” For more information about the PCI configurations, see Section 16.3.2.19, “PCI Bus Function Register (PBFR).” Figure 18-5 describes the bit settings of the PORBMSR.

Table 18-5. PORBMSR Field Descriptions

Bits	Name	Description
0	BCFG	CPU boot configuration 0 The CPU is prevented from booting until configuration by an external master is complete. 1 The CPU is allowed to start fetching boot code.
1–4	—	Reserved
5–7	ROM_LOC	Location of boot ROM: 000 PCI/PCI-X 001 DDR SDRAM 011 RapidIO 101 Local bus GPCM: 8-bit 110 Local bus GPCM:16-bit 111 Local bus GPCM: 32-bit
8–9	—	Reserved
10–11	BSCFG	Boot sequencer configuration 00 Reserved 01 Boot sequencer enabled with normal I^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 MPC8540 is an agent on an interface, it is prevented from mastering transactions on that interface until the external host configures the interface appropriately. 00 PCI and RapidIO agent mode 01 RapidIO agent mode 10 PCI/PCI-X agent mode 11 Host mode
16–31	—	Reserved

18.4.1.3 POR I/O Impedance Status and Control Register (PORIMPSCR)

PORIMPSCR, shown in Figure 18-3,contains the current I/O driver impedances for local bus and PCI interfaces.

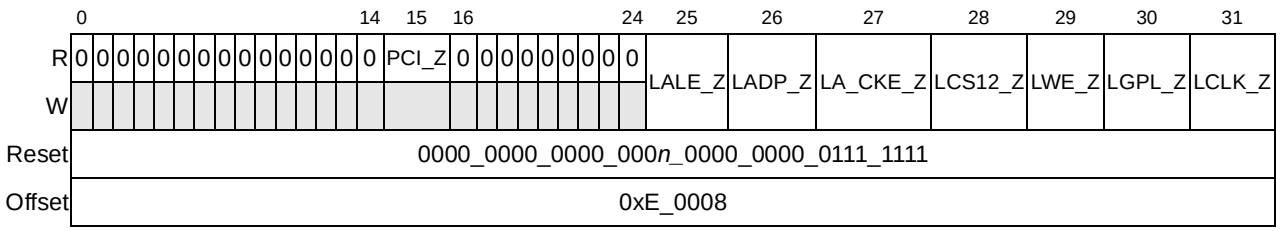


Figure 18-3. POR I/O Impedance Status and Control Register (PORIMPSCR)

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 *MPC8540 Integrated Processor Hardware Specifications* provides exact I/O impedances.

Table 18-6 describes PORIMPSCR fields.

Table 18-6. PORIMPSCR Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	PCI_Z	PCI/PCI-X I/O impedance 0 Low impedance 1 High impedance
16–24	—	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

18.4.1.4 POR Device Status Register (PORDEVSR)

Shown in Figure 18-4, 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.”

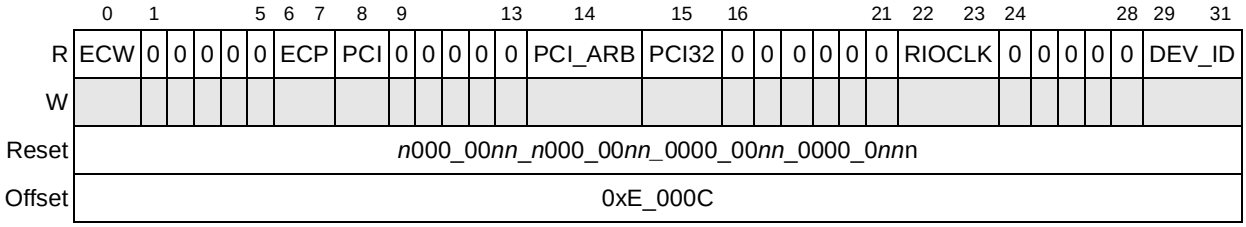


Figure 18-4. POR Device Status Register (PORDEVSR)

Table 18-7 describes the bit settings of PORDEVSR.

Table 18-7. PORDEVSR Field Descriptions

Bits	Name	Description
0	ECW	Gigabit Ethernet controller width 0 Reduced (RGMII, RTBI) 1 Full (MII, GMII, TBI)
1–5	—	Reserved
6-7	ECP	Gigabit Ethernet controller protocol 00 Both TSEC blocks use a Media Independent Interface protocols (MII/GMII/RGMII) 01 TSEC1 uses MII, TSEC2 uses TBI 10 TSEC1 uses TBI, TSEC2 uses MII 11 Both TSEC blocks use a ten bit interface protocol (TBI or RTBI)
8	PCI	PCI/PCI-X mode 0 PCI-X mode 1 PCI mode
9–13	—	Reserved
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]

18.4.1.5 POR Debug Mode Status Register (PORDBGMSR)

PORDBGMSR, shown in Figure 18-5, 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."

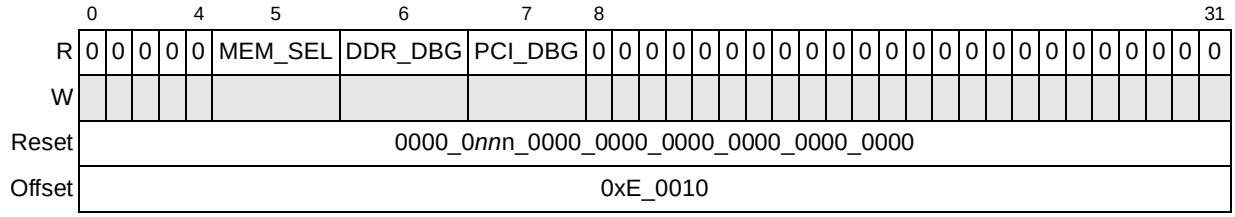


Figure 18-5. POR Debug Mode Status Register (PORDBGMSR)

Table 18-8 describes the bit settings of PORDBGMSR.

Table 18-8. PORDBGMSR Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5	MEM_SEL	Memory select. Indicates which 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

18.4.1.6 General-Purpose POR Configuration Register (GPPORCR)

Shown in Figure 18-6, 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.

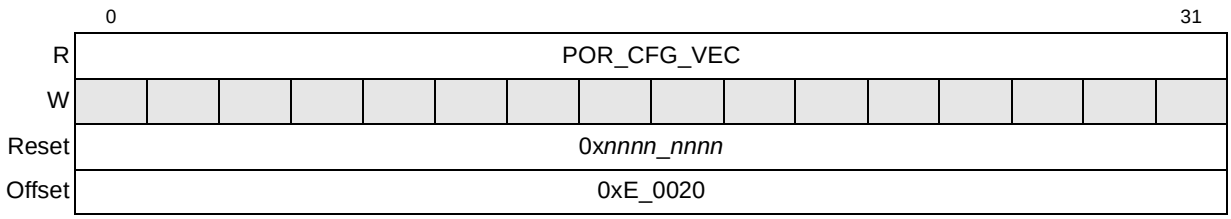


Figure 18-6. POR Configuration Register (GPPORCR)

Table 18-9 describes the bit settings of GPPORCR.

Table 18-9. GPPORCR Field Descriptions

Bits	Name	Description
0–31	POR_CFG_VEC	General-purpose POR configuration vector sampled from local bus address/data signals at the negation of HRESET. Note that if nothing is driven on these signals during reset, the value of this register is indeterminate.

18.4.1.7 General-Purpose I/O Control Register (GPIOCR)

Shown in Figure 18-7, 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 18.4.1.9, "General-Purpose Input Data Register (GPINDR)," and Section 18.4.1.8, "General-Purpose Output Data Register (GPOUTDR)." Section 18.5.2, "General-Purpose I/O Signals," describes the use of general-purpose I/O signals.

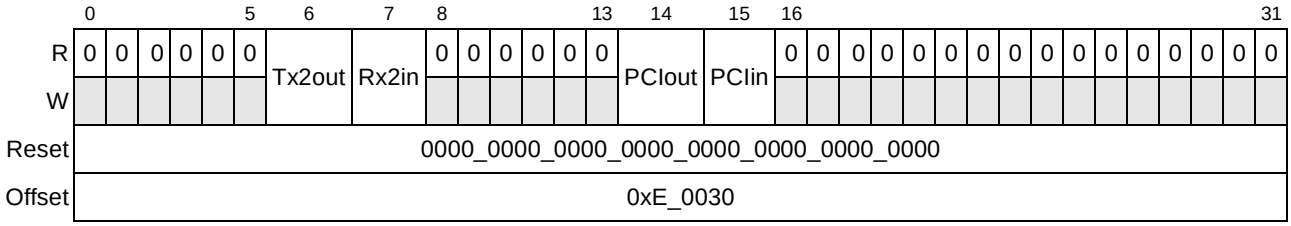


Figure 18-7. General-Purpose I/O Control Register (GPIOCR)

Table 18-10 describes the bit settings of GPIOCR.

Table 18-10. GPIOCR Field Descriptions

Bits	Name	Description
0–5	—	Reserved
6	Tx2out	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 18-10. 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

18.4.1.8 General-Purpose Output Data Register (GPOUTDR)

GPOUTDR, shown in Figure 18-8, 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 18.4.1.7, "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.

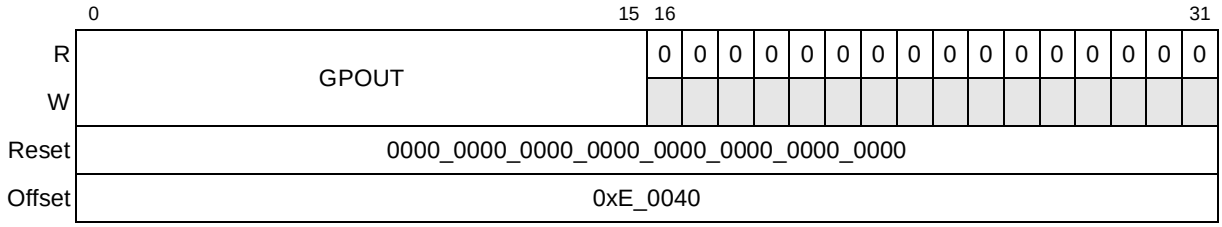


Figure 18-8. General-Purpose Output Data Register (GPOUTDR)

Table 18-11 describes the fields of GPOUTDR.

Table 18-11. GPOUTDR Field Descriptions

Bits	Name	Description
0–15	GPOUT	General-purpose output data. When the corresponding signals are configured to be general-purpose output signals, the values of the bits of GPOUT are driven onto those pins. GPOUTDR[0:7] corresponds to TSEC2_TxD[0:7]. GPOUTDR[8:15] corresponds to PCI_AD[47:40] as follows: GPOUTDR[8] ↔ PCI_AD[47] GPOUTDR[9] ↔ PCI_AD[46] GPOUTDR[10] ↔ PCI_AD[45] GPOUTDR[11] ↔ PCI_AD[44] GPOUTDR[12] ↔ PCI_AD[43] GPOUTDR[13] ↔ PCI_AD[42] GPOUTDR[14] ↔ PCI_AD[41] GPOUTDR[15] ↔ PCI_AD[40]
16–31	—	Reserved, should be cleared

18.4.1.9 General-Purpose Input Data Register (GPINDR)

GPINDR, shown in Figure 18-9, 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 18.4.1.7, "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.

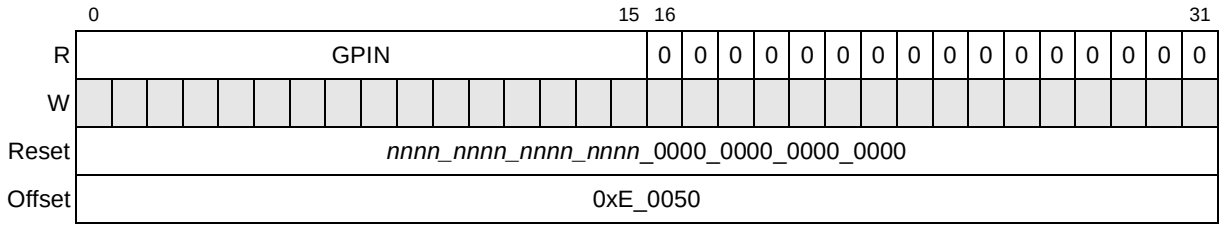


Figure 18-9. General-Purpose Output Data Register (GPINDR)

Table 18-11 describes the fields of GPINDR.

Table 18-12. GPINDR Field Descriptions

Bits	Name	Description
0–15	GPIN	General-purpose input data. When the corresponding signals are configured to be general-purpose input signals, the values sampled on these signals are reflected in GPIN. GPINDR[0:7] corresponds to TSEC2_RxD[0:7]. GPINDR[8:15] corresponds to PCI_AD[39:32] as follows: GPINDR[8] ↔ PCI_AD[39] GPINDR[9] ↔ PCI_AD[38] GPINDR[10] ↔ PCI_AD[37] GPINDR[11] ↔ PCI_AD[36] GPINDR[12] ↔ PCI_AD[35] GPINDR[13] ↔ PCI_AD[34] GPINDR[14] ↔ PCI_AD[33] GPINDR[15] ↔ PCI_AD[32]
16–31	—	Reserved

18.4.1.10 Alternate Function Signal Multiplex Control Register (PMUXCR)

Shown in Figure 18-10, 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.

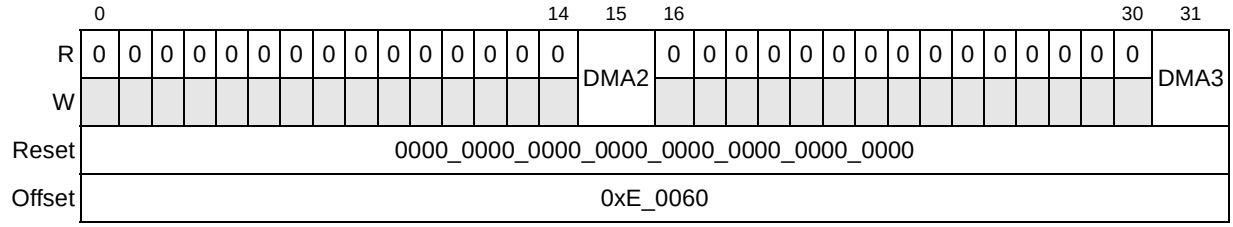


Figure 18-10. Alternate Function Pin Multiplex Control Register (PMUXCR)

Table 18-13 describes the bit settings of PMUXCR.

Table 18-13. PMUXCR Field Descriptions

Bits	Name	Description
0–14	—	Reserved
15	DMA2	Enables DMA channel 2 signals 0 DMA channel 2 is disabled; the pins retain their primary function as local bus chip selects 1 DMA channel 2 is enabled and $\overline{\text{LCS5}}$ functions as $\overline{\text{DMA\_DREQ2}}$ $\overline{\text{LCS6}}$ functions as $\overline{\text{DMA\_DACK2}}$ $\overline{\text{LCS7}}$ 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 $\overline{\text{DMA\_DREQ3}}$ IRQ10 functions as $\overline{\text{DMA\_DACK3}}$ IRQ11 functions as DMA_DDONE3

18.4.1.11 Device Disable Register (DEVDISR)

DEVDISR, shown in Figure 18-11, contains disable bits for various MPC8540 functional blocks.

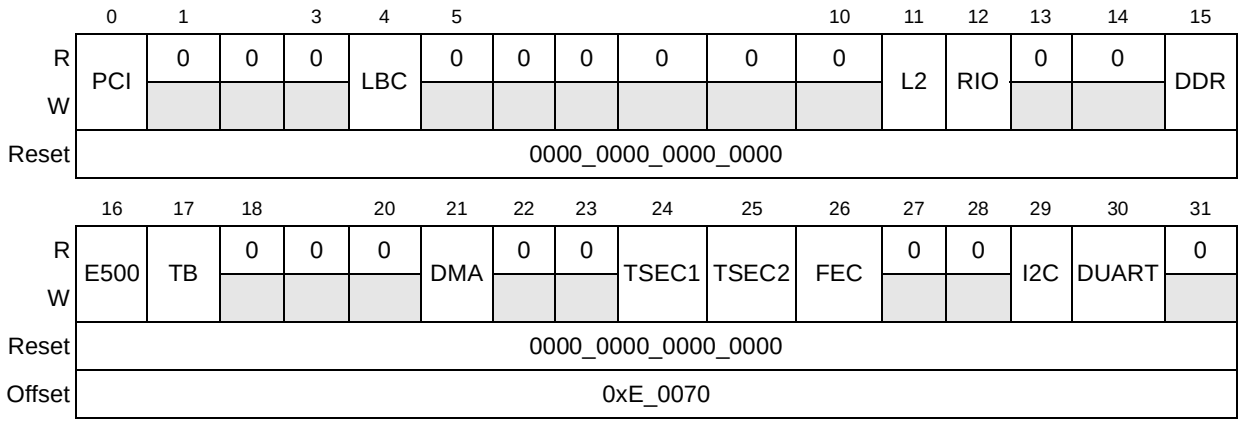


Figure 18-11. Device Disable Register (DEVDISR)

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 18.4.1.7, “General-Purpose I/O Control Register (GPIOCR).” Blocks disabled by DEVDISR must not be re-enabled without a hard reset. Section 18.5.1.4, “Shutting Down Unused Blocks,” has more information on the use of DEVDISR. Table 18-14 describes DEVDISR fields.

Table 18-14. 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 18.5.1.4, “Shutting Down Unused Blocks.”
17	TB	Time base (timer facilities) of the e500 core disable 0 Timer facilities enabled 1 Timer facilities disabled
18–20	—	Reserved
21	DMA	DMA controller disabled 0 DMA controller enabled 1 DMA controller disabled
22–23	—	Reserved
24	TSEC1	Three-speed Ethernet controller 1 disable 0 TSEC1 enabled 1 TSEC1 disabled

Table 18-14. DEVDISR Field Descriptions (continued)

Bits	Name	Description
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	FEC	Fast Ethernet controller (10/100 maintenance port) disable 0 FEC enabled 1 FEC disabled
27–28	—	Reserved
29	I2C	I^2C controller disabled 0 I^2C controller enabled 1 I^2C controller disabled
30	DUART	Dual UART controller disabled 0 DUART enabled 1 DUART disabled
31	—	Reserved

18.4.1.12 Power Management Control and Status Register (POWMGTCSR)

Shown in Figure 18-12, POWMGTCSR contains bits for placing the MPC8540 into low power states and for controlling when it wakes up. It also contains power management status bits. See Section 18.5.1.8.2, "Interrupts and Power Management Controlled by POWMGTCSR," for more information.

	0	1	2–11	12	13	14	15–27	28	29	30	31
R	IRQ_MSK	CI_MSK	0	DOZ	0	SLP	0	DOZING	NAPPING	SLPING	0
W											
Reset	0000_0000_0000_0000_0000_0000_0000_0000										
Offset	0xE_0080										

Figure 18-12. Power Management Control and Status Register (POWMGTCSR)

Table 18-15 describes the bit settings of POWMGTCSR.

Table 18-15. 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.

Table 18-15. POWMGTCSR Field Descriptions (continued)

Bits	Name	Description
2–11	—	Reserved
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 MPC8540 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 MPC8540 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.

18.4.1.13 Machine Check Summary Register (MCPSUMR)

Shown in Figure 18-13, 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).”

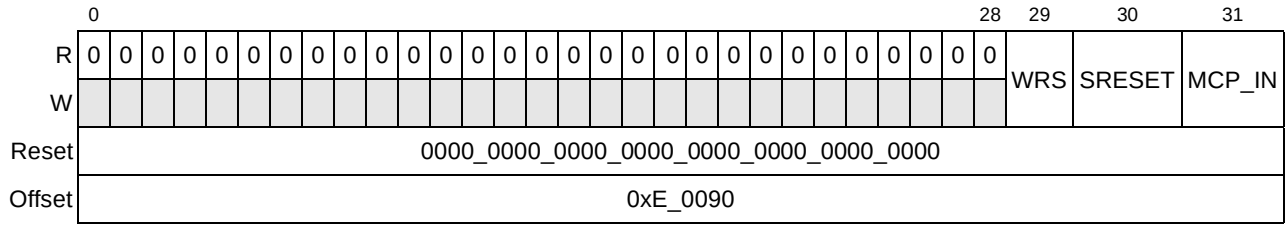


Figure 18-13. Machine Check Summary Register (MCPSUMR)

Table 18-16 describes the bit settings of MCPSUMR.

Table 18-16. MCPSUMR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29	WRS	Watchdog timer 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

18.4.1.14 Processor Version Register (PVR)

Shown in Figure 18-14, 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 MPC8540.

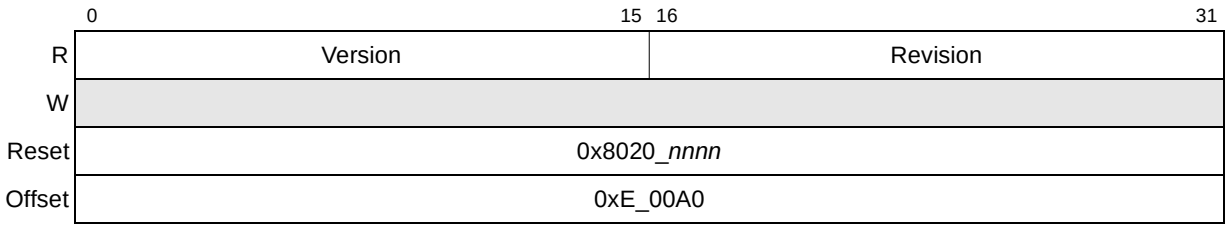


Figure 18-14. Processor Version Register (PVR)

Table 18-17 describes the fields of PVR.

Table 18-17. PVR Field Descriptions

Bits	Name	Description
0–15	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.

18.4.1.15 System Version Register (SVR)

Shown in Figure 18-15, the SVR contains the system version number for the MPC8540 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 MPC8540.

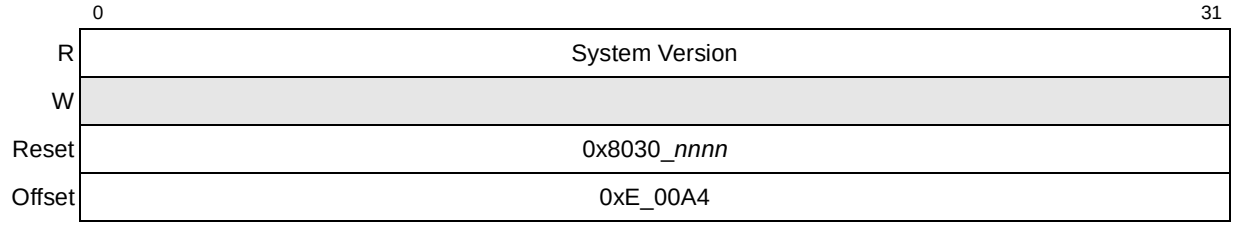


Figure 18-15. System Version Register (SVR)

Table 18-18 describes the fields of SVR.

Table 18-18. SVR Field Descriptions

Bits	Name	Description
0–31	SV	System version for the MPC8540 system logic

18.4.1.16 Clock Out Control Register (CLKOCR)

Shown in Figure 18-16, the CLKOCR contains control bits that select the clock sources to be placed on the clock out (CLK_OUT) signal.

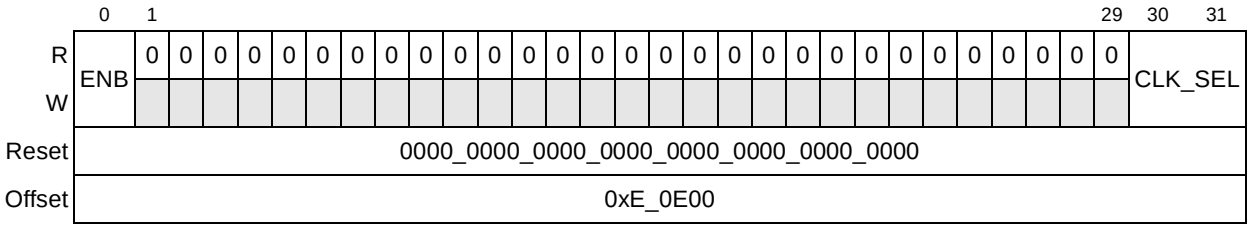


Figure 18-16. Clock Out Control Register (CLKOCR)

Table 18-19 describes the bit settings of CLKOCR.

Table 18-19. CLKOCR Field Descriptions

Bits	Name	Description
0	ENB	Clock out enable 0 CLK_OUT signal is tri-stated 1 CLK_OUT signal is driven according to CLKOCR[CLK_SEL]
1–29	—	Reserved
30–31	CLK_SEL	Clock out select 00 CCB (platform) clock 01 CCB (platform) clock divided by 2 10 SYSCLK (echoes SYSCLK input) 11 SYSCLK divided by 2 (demonstrates platform PLL lock)

18.4.1.17 DDR DLL Control Register (DDRDLLCR)

The DDRDLLCR, shown in Figure 18-17, contains bits that allow control and debug of the DDR SDRAM controller's DLL. The DLL delay chain consists of 128 tap points.

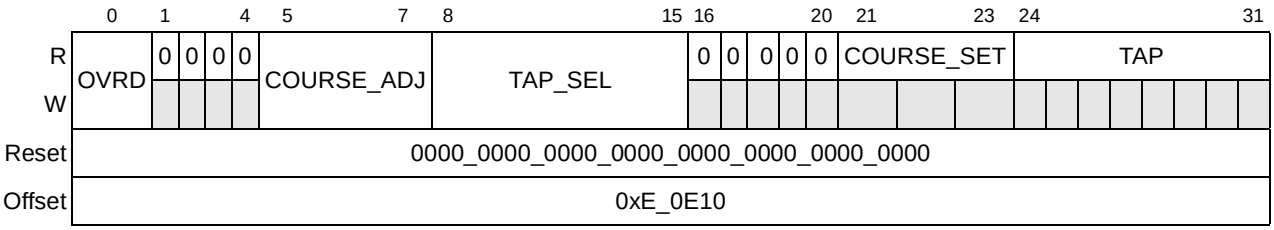


Figure 18-17. DDR DLL Control Register (DDRDLLCR)

Table 18-20 describes the bit settings of DDRDLLCR.

Table 18-20. DDRDLLCR Field Descriptions

Bits	Name	Description
0	OVRD	Override mode 0 Override mode disabled 1 Override of current delay chain tap point with the TAPSEL tap point enabled
1–4	—	Reserved
5–7	COURSE_ADJ	Course adjustment value to be used by the DLL when in override mode (OVRD = 1). The course adjustment is the number of CCB (platform) clock cycles of delay to inject before the delay chain. When leaving override mode (OVRD cleared), this course adjustment point serves as the starting point for a dynamic search for a lock point.
8–15	TAP_SEL	TAP select value to be used by the DLL when in override mode (OVRD = 1). Selects the tap point within the delay chain. When leaving override mode (OVRD cleared) this tap point serves as the starting point for a dynamic search for a lock point.
16–20	—	Reserved

Table 18-20. 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

18.4.1.18 Local Bus DLL Control Register (LBDLLCR)

Shown in Figure 18-18, the LBDLLCR contains control bits that allow debug of the local bus controller's DLL. The delay chain of the DLL is made up of 128 tap points.

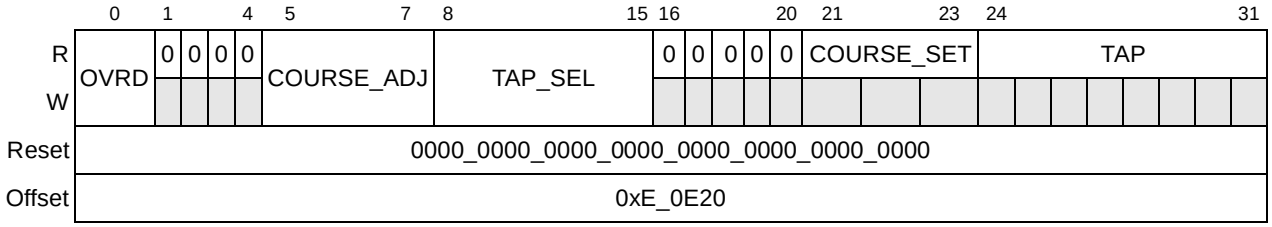


Figure 18-18. Local Bus DLL Control Register (LBDLLCR)

Table 18-21 describes the bit settings of LBDLLCR.

Table 18-21. LBDLLCR Field Descriptions

Bits	Name	Description
0	OVRD	Override mode 0 Override mode disabled 1 Override of current delay chain tap point with the TAPSEL tap point enabled
1–4	—	Reserved
5–7	COURSE_ADJ	Course adjustment value to be used by the DLL when in override mode (OVRD = 1). The course adjustment is the number of CCB clock cycles of delay to inject before the delay chain. When leaving override mode (OVRD cleared) this course adjust point serves as the starting point for a dynamic search for a lock point.
8–15	TAP_SEL	TAP select value to be used by the DLL when in override mode (OVRD = 1). Selects the tap point within the delay chain. When leaving override mode (OVRD cleared) this tap point serves as the starting point for a dynamic search for a lock point.
16–20	—	Reserved
21–23	COURSE_SET	Reports the current course delay setting found by the dynamic search algorithm that produced a lock. Measured in CCB clock cycles
24–31	TAP	Reports the tap value found by the dynamic search algorithm that produced a lock. Measured in tap points

18.5 Functional Description

This section describes the global utilities from a functional perspective.

18.5.1 Power Management

The MPC8540 has features to minimize power consumption at several levels. Dynamic power management locally minimizes power consumption when a block is idle. Software can also shut down clocks to individual blocks when they are not needed 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 MPC8540 memory-mapped registers through the RapidIO or PCI interfaces.

These features are described in further detail in this section.

18.5.1.1 Relationship Between Core and Device Power Management States

The MPC8540 has three low-power states: doze, nap, and sleep. The mapping of core and device power management states is shown in Figure 18-19 showing state transitions from the perspective of the e500 core.

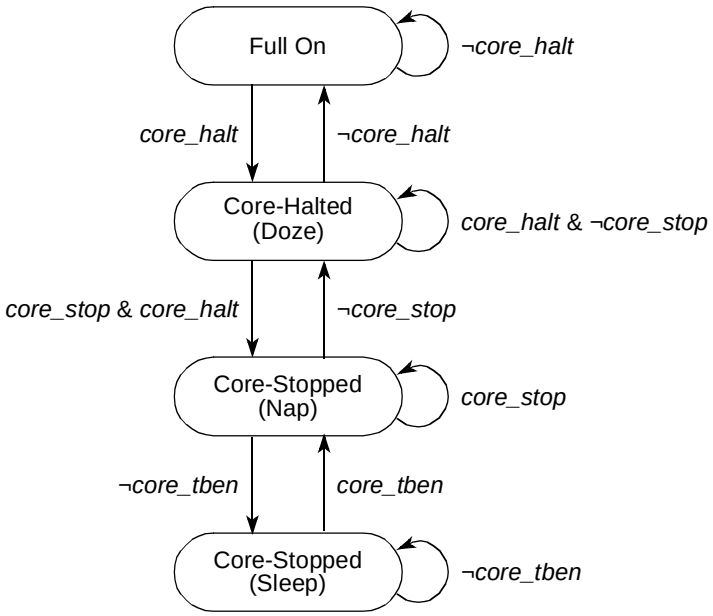


Figure 18-19. e500 Core Power Management State Diagram

For each operating state represented in the diagram, the core's state is listed first, with the corresponding state of the MPC8540 shown beneath it in parenthesis. Note that there are many other variables that control the state transitions between MPC8540 power management states. These additional variables are described in more detail in Section 18.5.1.7, "Power-Down Sequence Coordination."

Table 18-22 lists basic characteristics of the low-power modes and the full on mode.

Table 18-22. MPC8540 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

18.5.1.2 $\overline{\text{CKSTP\_IN}}$ 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 18.3.2, "Detailed Signal Descriptions."

18.5.1.3 Dynamic Power Management

Many blocks in the MPC8540 can dynamically turn off clocks within the block when sections of the block are idle. This feature is always enabled and occurs automatically.

18.5.1.4 Shutting Down Unused Blocks

As described in Section 18.4.1.11, "Device Disable Register (DEVDISR)," DEVDISR provides a way to shut down certain functional blocks within the MPC8540 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.

18.5.1.5 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 MPC8540 to cause the device to enter doze or sleep modes.

18.5.1.5.1 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 MPC8540 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 18.5.1.8, “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.

18.5.1.5.2 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 MPC8540 returns to nap mode after the interrupt is serviced. See Section 18.5.1.8, “Interrupts and Power Management,” for more information.

All device logic external to the e500 core remains fully operational in nap mode.

18.5.1.5.3 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 MPC8540 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.

After the core and I/O interfaces have shut down, ASLEEP is asserted and READY is negated.

NOTE

Only external interrupts can wake the MPC8540 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.

18.5.1.6 Power Management Control Fields

The e500 core provides the following fields to signal power management requests to the MPC8540 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 MPC8540 to initiate doze mode
- HID0[NAP]—Signals the MPC8540 to initiate nap mode
- HID0[SLEEP]—Signals the MPC8540 to initiate sleep mode

These register fields and their functional relationship are shown in Figure 18-20. 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.

18.5.1.7 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 18-20, and protocols with all other MPC8540 functional blocks that respond to power-down requests. The mode-transition protocol is executes automatically under these conditions and is shown in Figure 18-19 and described in Table 18-23.

The column in Table 18-23 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 MPC8540, 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 18-23. 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 18-20, the e500 core enters low-power modes only in response to the *core_halt*, *core_stop*, or *core_tben* inputs from the MPC8540'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 18-20 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)."

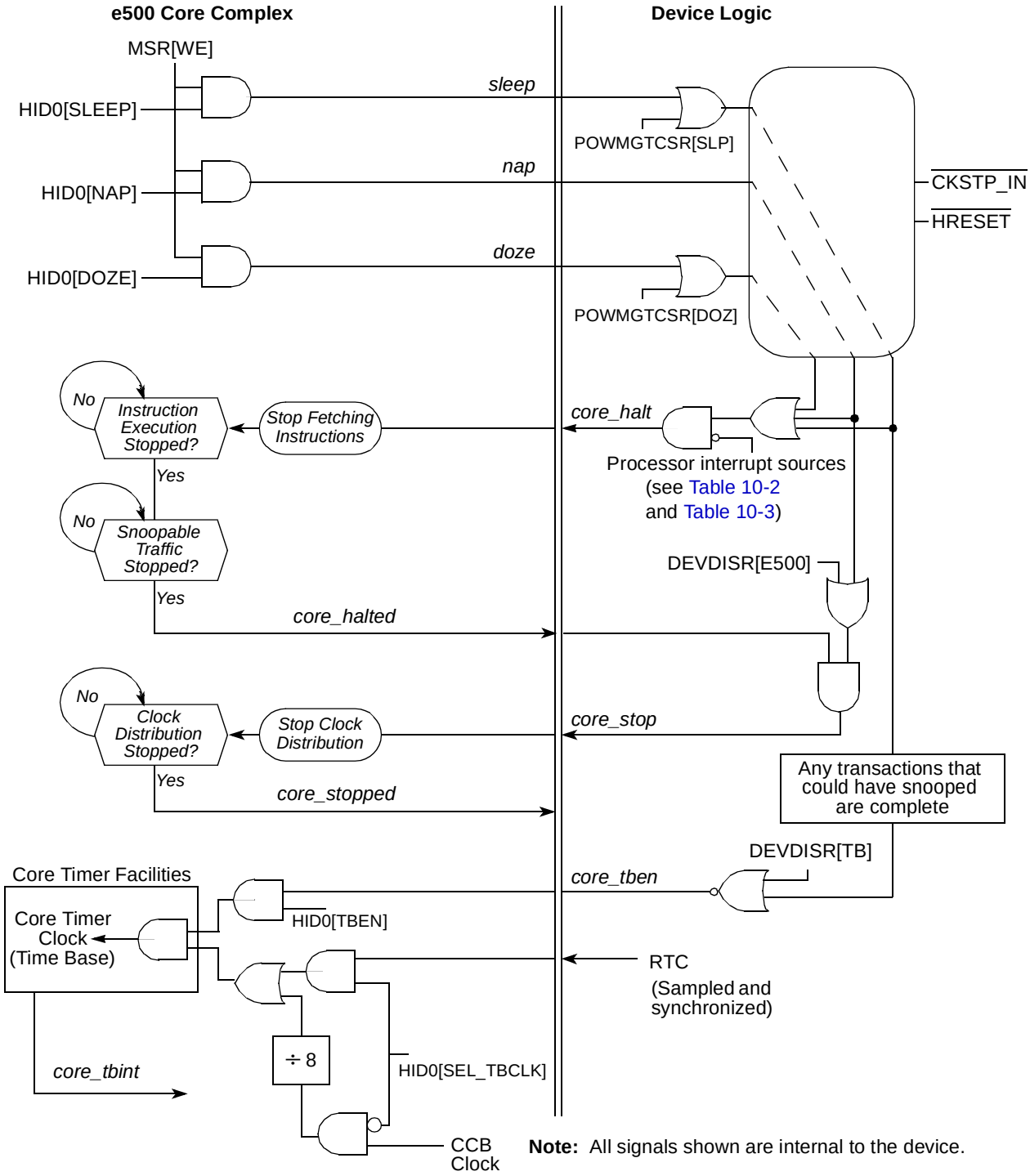


Figure 18-20. MPC8540 Power Management Handshaking Signals

18.5.1.8 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.

18.5.1.8.1 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 MPC8540 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 MPC8540 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.

18.5.1.8.2 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 18.4.1.12, "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.

18.5.1.9 Snooping in Power-Down Modes

When the MPC8540 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 MPC8540 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 MPC8540 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.

18.5.1.10 Software Considerations for Power Management

Setting MSR[WE] generates a request to the MPC8540 logic (external to the core complex) to enter a power saving state. 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
```

18.5.1.11 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, FEC, 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.

18.5.2 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 18.4.1.7, "General-Purpose I/O Control Register (GPIOCR)."

18.5.3 Interrupt and Local Bus Signal Multiplexing

The MPC8540 has very little signal multiplexing. Two sets of DMA channel triggering signals can alternately be placed on other signals as follows:

- $\overline{\text{LCS}}$[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 18.4.1.10, "Alternate Function Signal Multiplex Control Register (PMUXCR)."

Chapter 19
Performance Monitor

This chapter describes the MPC8540 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 19.4.7, "Performance Monitor Events," briefly describes the events that can be monitored. Refer to the individual chapters for a better understanding of these events.

19.1 Introduction

The MPC8540 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 MPC8540's behavior in any system or software environment.
- The performance monitor facility can be used to aid system developers when bringing up and debugging systems.
- System performance can be increased by monitoring memory hierarchy behavior. This can help to optimize algorithms used to schedule or partition tasks and to refine the data structures and distribution used by each task.

19.1.1 Overview

Figure 19-1 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.

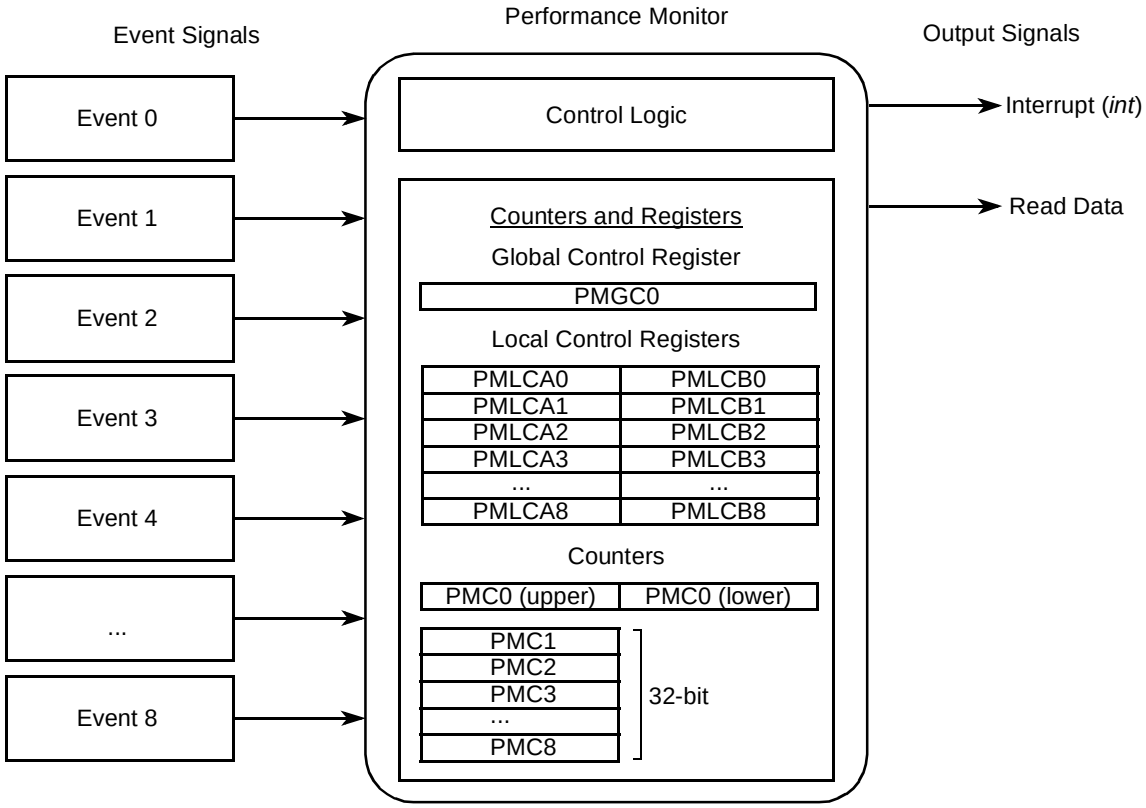


Figure 19-1. Performance Monitor Block Diagram

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 19-1.

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.

19.1.2 Features

The MPC8540 performance monitor offers a rich set of features that permits a complete performance characterization of the implementation. These features include:

- One 64-bit counter exclusively dedicated to counting cycles
- Eight 32-bit counters that count the occurrence of selected events
- One global control register (affects all counters) and two local control registers per counter
- Ability to count up to 64 reference events that may be counted on any of the eight 32-bit counters
- Ability to count up to 512 counter-specific events
- Triggering and chaining capability
- Duration threshold counting
- Burstiness feature that permits counting of burst events with a programmable time between bursts
- Ability to generate an interrupt on overflow

19.2 External 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.

19.3 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 19-1 lists the performance monitor registers. These registers can be read or written only with 32-bit accesses.

19.3.1 Register Summary

The performance monitor uses nine counter registers and a group of local control registers that are used to specify the method of counting. Two local control registers are associated with each counter in addition to a global control register that applies to all counters.

Table 19-1. 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	19.3.2.1/19-5
0xE_1010	PMLCA0—Performance monitor local control register A0	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1014	PMLCB0—Performance monitor local control register B0	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1018	PMC0 (upper)—Performance monitor counter 0 upper	R/W	0x0000_0000	19.3.3.1/19-9
0xE_101C	PMC0 (lower)—Performance monitor counter 0 lower	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1020	PMLCA1—Performance monitor local control register A1	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1024	PMLCB1—Performance monitor local control register B1	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1028	PMC1—Performance monitor counter 1	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1030	PMLCA2—Performance monitor local control register A2	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1034	PMLCB2—Performance monitor local control register B 2	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1038	PMC2—Performance monitor counter 2	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1040	PMLCA3—Performance monitor local control register A3	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1044	PMLCB3—Performance monitor local control register B3	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1048	PMC3—Performance monitor counter 3	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1050	PMLCA4—Performance monitor local control register A4	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1054	PMLCB4—Performance monitor local control register B4	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1058	PMC4—Performance monitor counter 4	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1060	PMLCA5—Performance monitor local control register A5	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1064	PMLCB5—Performance monitor local control register B 5	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1068	PMC5—Performance monitor counter 5	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1070	PMLCA6—Performance monitor local control register A6	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1074	PMLCB6—Performance monitor local control register B6	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1078	PMC6—Performance monitor counter 6	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1080	PMLCA7—Performance monitor local control register A7	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1084	PMLCB7—Performance monitor local control register B7	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1088	PMC7—Performance monitor counter 7	R/W	0x0000_0000	19.3.3.1/19-9
0xE_1090	PMLCA8—Performance monitor local control register A8	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1094	PMLCB8—Performance monitor local control register B8	R/W	0x0000_0000	19.3.2.2/19-5
0xE_1098	PMC8—Performance monitor counter 8	R/W	0x0000_0000	19.3.3.1/19-9

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).”

19.3.2 Control Registers

This section describes the performance monitor control registers in detail.

19.3.2.1 Performance Monitor Global Control Register (PMGC0)

The performance monitor global control register (PMGC0), shown in Figure 19-2, is a 32-bit register used to control all PMCs.

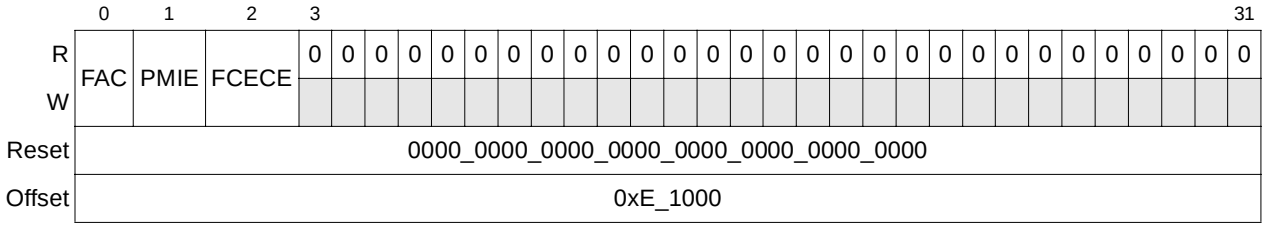


Figure 19-2. Performance Monitor Global Control Register (PMGC0)

Table 19-2 describes PMGC0 fields.

Table 19-2. 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

19.3.2.2 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 19-3 shows the performance monitor local control A0 register (PMLCA0).

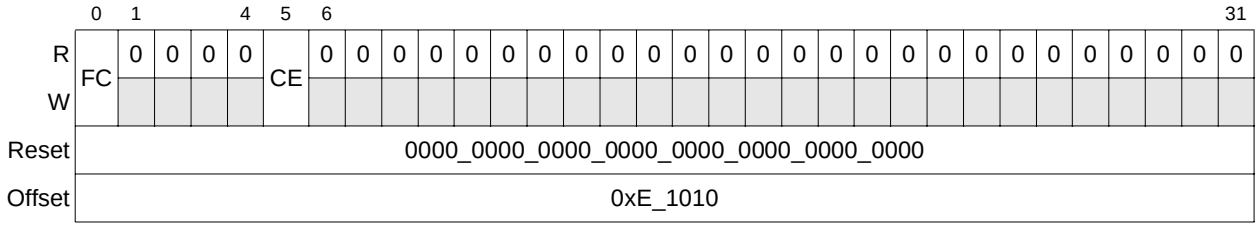


Figure 19-3. Performance Monitor Local Control Register A0 (PMLCA0)

Table 19-3 describes PMLCA0 fields.

Table 19-3. 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 19-4 shows the performance monitor local control registers A1–A8.

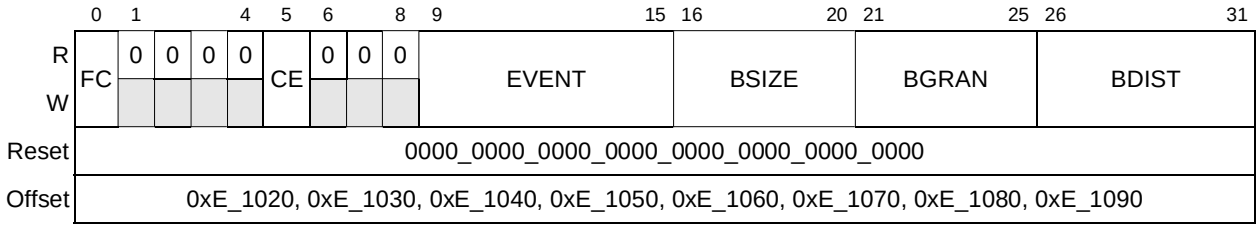


Figure 19-4. Performance Monitor Local Control Registers A1–A8 (PMLCA1–PMLCA8)

Table 19-4 describes PMLCA*n* fields.

Table 19-4. 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 19-4. 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 19-10 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 19.4.6, “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 19.4.6, “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 19-5 shows the performance monitor local control B0 register (PMLCB0).

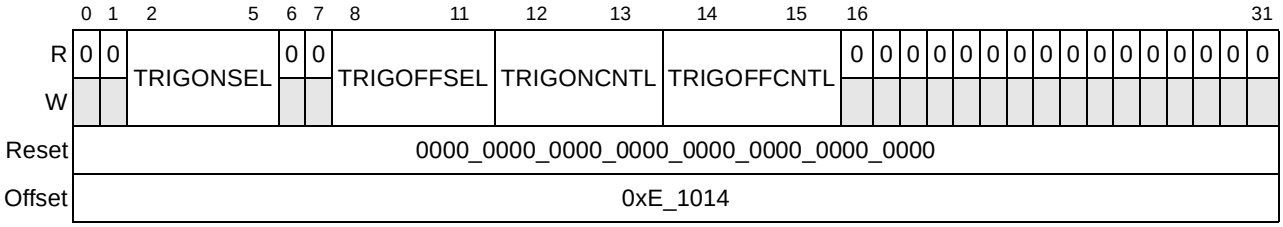


Figure 19-5. Performance Monitor Local Control Register B0 (PMLCB0)

Table 19-5 describes PMLCB0 fields.

Table 19-5. 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 19-5. 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 19-6 shows performance monitor local control registers B1–B8.

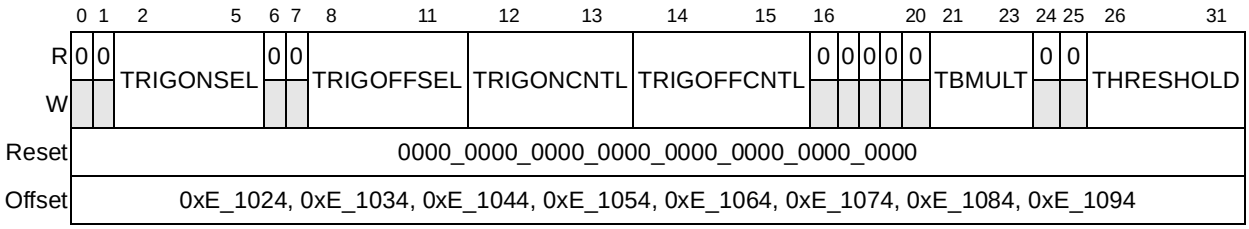


Figure 19-6. Performance Monitor Local Control Registers B1–B8 (PMLCB1–PMLCB8)

Table 19-5 describes PMLCB*n* fields.

Table 19-6. 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 19-6. 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.

19.3.3 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.

19.3.3.1 Performance Monitor Counters (PMC0–PMC8)

PMC0–PMC8 are used to count events selected by the performance monitor local control registers. PMC0, shown in Figure 19-7, 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.

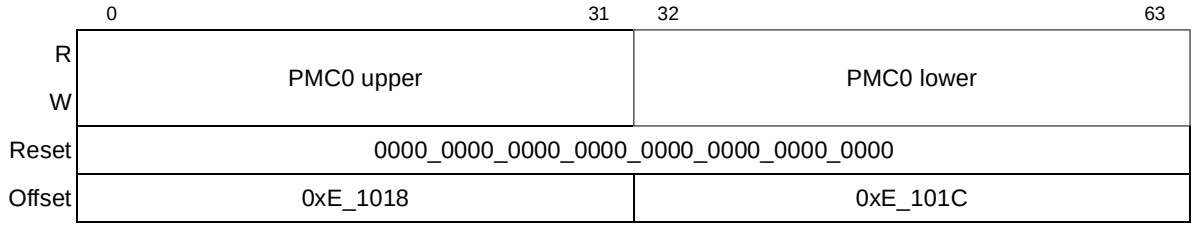


Figure 19-7. Performance Monitor Counter Register 0 (PMC0)

Table 19-7 describes PMC0 fields.

Table 19-7. PMC0 Field Descriptions

Bits	Name	Description
0–63	PMC0	Event count. Counts only clock cycles

PMC1–PMC8, shown in Figure 19-8, 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.

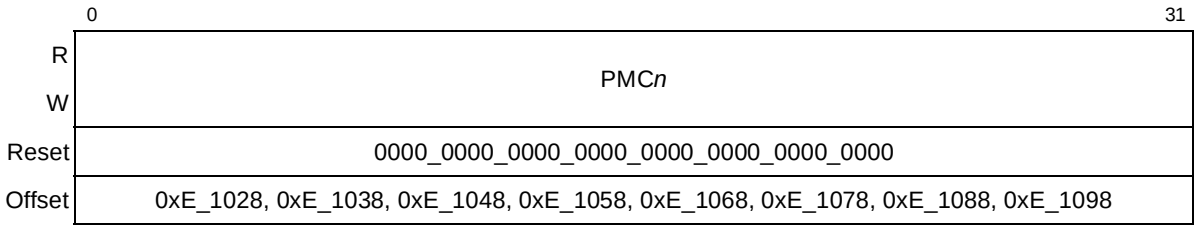


Figure 19-8. Performance Monitor Counter Registers 1–8 (PMC1–PMC8)

Table 19-8 describes PMC*n* fields.

Table 19-8. 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.

19.4 Functional Description

This section describes the use of some features of the performance monitor.

19.4.1 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.

19.4.2 Event Counting

Using the control registers described in Section 19.3.2, “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.

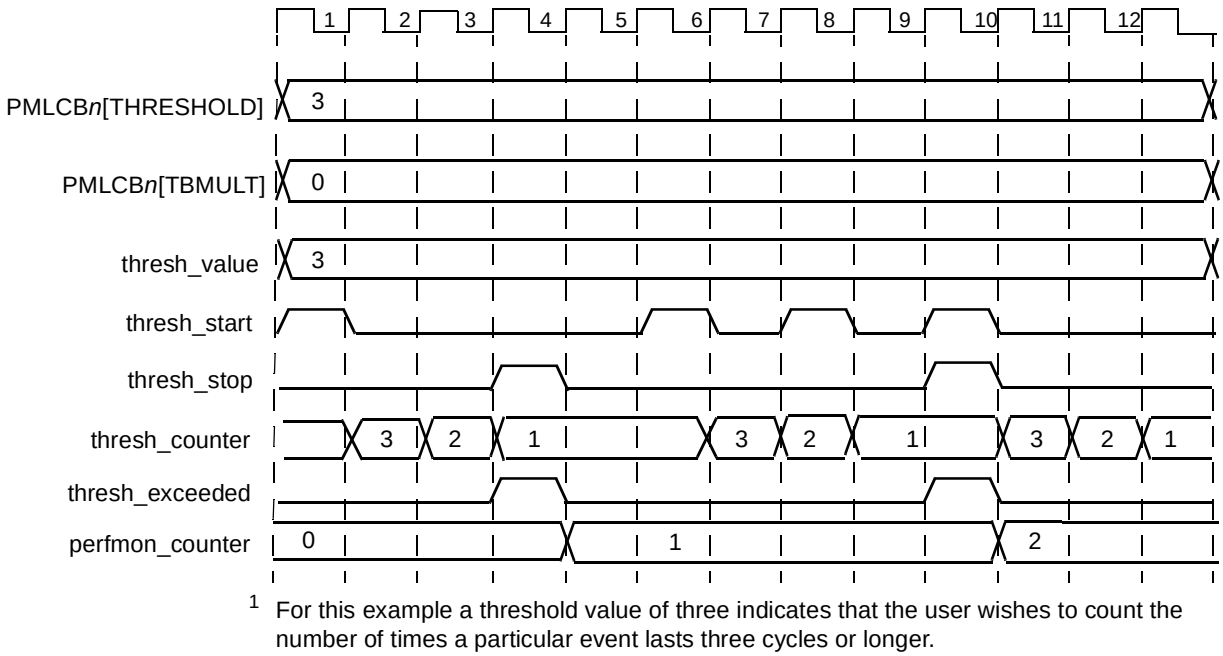
19.4.3 Threshold Events

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 19-9 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 19-9. Duration Threshold Event Sequence Timing Diagram

19.4.4 Chaining

By configuring one counter to increment each time another counter overflows, several counters can be chained together to provide event counts larger than 32 bits. Each counter in a chain adds 32 bits to the maximum count. The register chaining sequence is not arbitrary and is specified indirectly by selecting the register overflow event to be counted. Selecting an event has the effect of selecting a source register because all available chaining events, as shown in Table 19-10, 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.

19.4.5 Triggering

Triggering allows one counter to start or stop counting on the change of another counter or on the overflow of another counter. More specifically, if PMC1 is set to start or stop counting as a result of a change or overflow in counter PMC2, then counter PMC2 must be identified in the local

control register of counter PMC1. This is 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.

19.4.6 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 19-9, event bursts are defined by size, granularity, and distance.

Table 19-9. 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 19-10 shows the relationships between size, granularity, and distance. Burstiness counting can be performed for all events except threshold events.

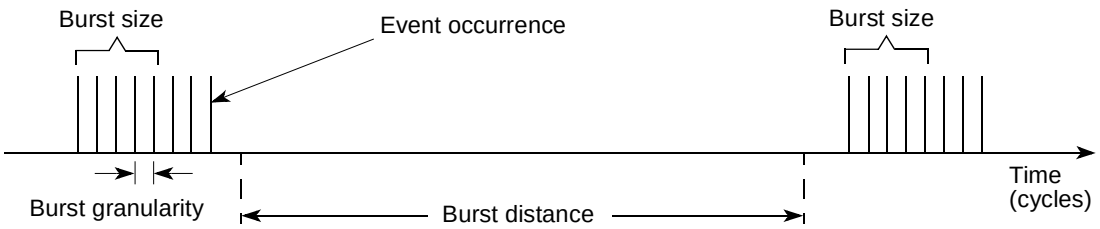


Figure 19-10. Burst Size, Distance, Granularity, and Burstiness Counting

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.

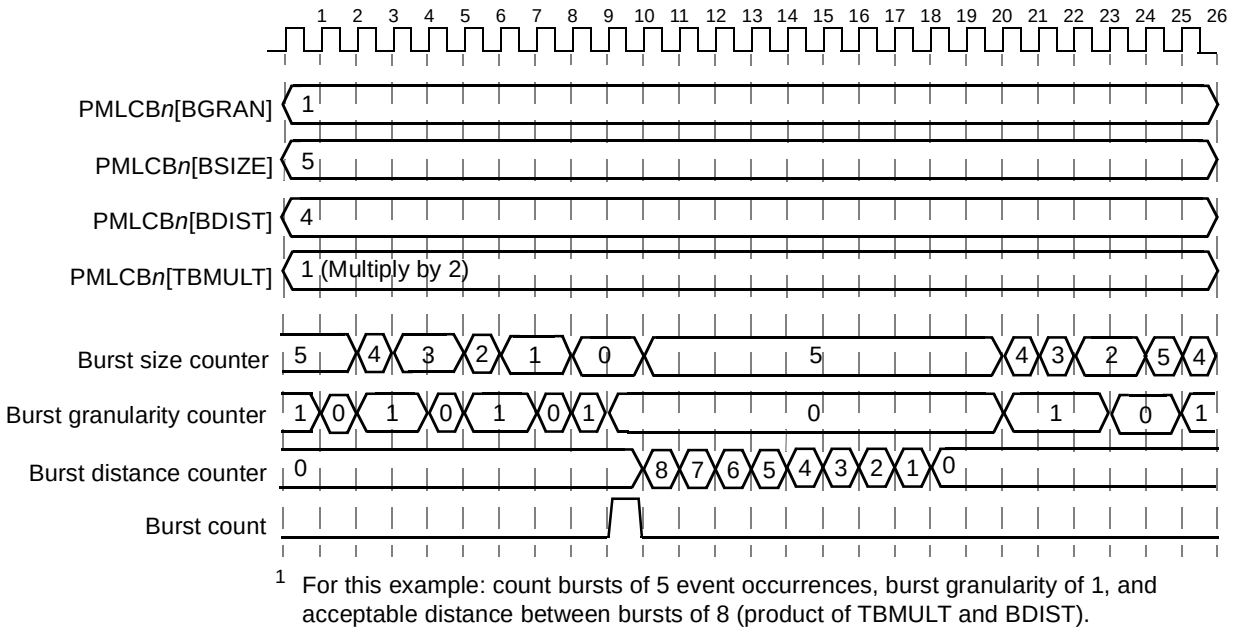
Burstiness counting is disabled when the definition of a burst is ambiguous, that is, when the burst size field is less than two, or the burst distance is zero. When burstiness counting is disabled, regular counting is allowed.

Figure 19-10 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 19-11 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 19-11. Burstiness Counting Timing Diagram

19.4.7 Performance Monitor Events

Table 19-10 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 19-10 are reserved.

Table 19-10. 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 19-10. 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 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
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 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
Inbound message packet protocol level retry	C4:18	—

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
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 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
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)

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
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 TSEC1	C3:23	—
ECM data bus grant TSEC2	C4:25	—
ECM data bus wait DDR	C5:20	—
ECM data bus wait LBC	C6:20	—
ECM data bus wait PIC	C5:21	—
ECM data bus wait TSEC1	C7:19	—
ECM data bus wait TSEC2	C8:18	—
ECM global data bus beat	Ref:16	—

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
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	—
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

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
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	—
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

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
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	—
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	—

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
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
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

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
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
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)

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
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	—
Bank 6 hits (chip-select)	C6:53	—
Bank 7 hits (chip-select)	C7:50	—
Bank 8 hits (chip-select)	C8:50	—
Requests granted to ECM port	C2:57	—
Cycles atomic reservation for ECM port is enabled	C4:55	

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
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	—
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	—

Table 19-10. Performance Monitor Events (continued)

Event Counted	Number	Description of Event Counted
Trace buffer hits	C1:58	—
DUART Events		
UART0 baud rate	C1:63	—
UART1 baud rate	C5:63	—
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.

19.4.8 Performance Monitor Examples

Table 19-12 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 19-11 are identical for all four examples.

Table 19-11. 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 19-12. Register Settings for Counting Examples MPC8540

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
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 20
Debug Features and Watchpoint Facility

This chapter describes all customer-visible debug modes of the MPC8540 integrated device. The debug features on the MPC8540 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 MPC8540 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.

20.1 Introduction

As shown in the block diagram of Figure 20-1, the MPC8540 device provides the following debug features (listed with references to sections of this chapter that describe them):

- PCI/PCI-X interface debug (Section 20.4.2, “PCI/PCI-X Interface Debug”)
- DDR SDRAM interface debug (Section 20.4.3, “DDR SDRAM Interface Debug”)
- Local bus controller (LBC) debug (Section 20.4.4, “Local Bus Interface Debug”)
- Watchpoint monitor and trace buffer debug (Section 20.4.5, “Watchpoint Monitor,” and Section 20.4.6, “Trace Buffer”)

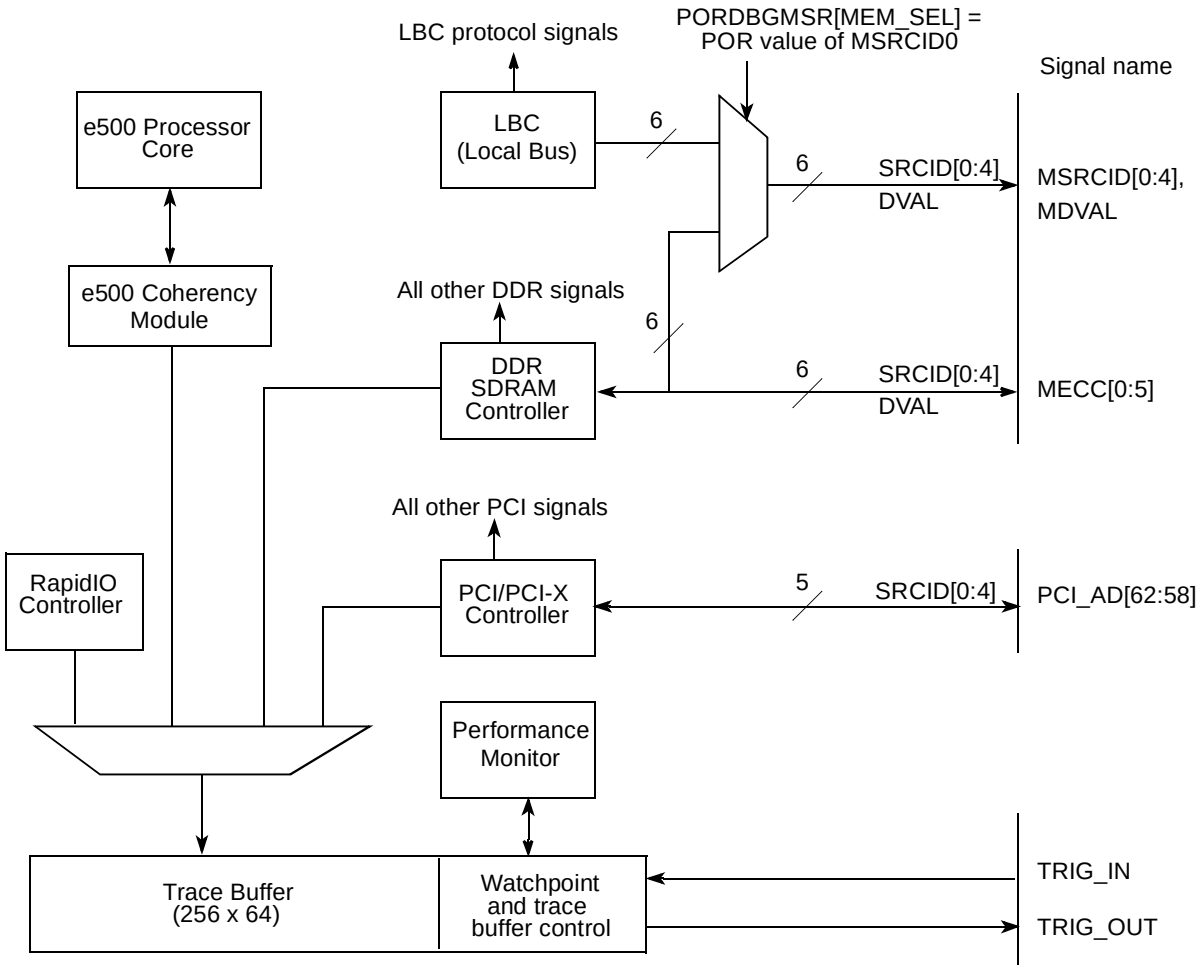


Figure 20-1. Debug and Watchpoint Monitor Block Diagram

20.1.1 Overview

As shown in Figure 20-1, 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.

Othmer 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 20.3.3, “Context ID Registers,” are useful for software debug.

20.1.2 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

20.1.3 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 20-3. The DDR controller can also drive debug information on either MSRCID[0:4] or

MECC[0:5]. See Section 20.4.1, “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 20-1. 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.	20.1.3.1/20-4
	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.	20.1.3.2/20-4
	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.	20.1.3.3/20-5
	1	Default value (internal pull-up resistor). The PCI/PCI-X interface operates in normal mode.	

20.1.3.1 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 20-1, 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.

20.1.3.2 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 20-1, 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.

20.1.3.3 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 20.4.2, "PCI/PCI-X Interface Debug," for more information.

20.1.3.4 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.

20.1.3.5 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.

20.2 External Signal Descriptions

This section provides information about all the external signals associated with the various MPC8540 debug functions.

As shown in Table 20-1, the MPC8540 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 MPC8540 provides a JTAG test access port (TAP) that complies with the IEEE 1149.1 boundary-scan specification. This section also describes JTAG TAP signals.

20.2.1 Overview

All the signals associated with device debug features are summarized in Table 20-2, listed with a reference to the page number of the section with more information. The detailed descriptions are contained in Table 20-2. 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 20-2. 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	20-7
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]	20-8
MSRCID[0:1]	Memory source ID	Debug	Selectable transaction source ID from either DDR SDRAM controller or local bus controller	Reset_cfg	O	20-8
MSRCID[2:4]				111	O	20-8
TRIG_IN	Trigger in	Debug	Trigger for various function in the watchpoint monitor and trace buffer	1	I	20-8
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 20-25).	1	O	20-8
PCI_AD[62:58]	PCI address	Debug	In debug mode these pins carry the source ID of the transaction.	0000_0	O	20-8
TCK	Test clock	Debug	Clock for JTAG testing. Internally pulled up	1	I	20-9
TDI	Test data input	Debug	Serial input for instructions and data to the JTAG test subsystem. Internally pulled up	1	I	20-9
TDO	Test data output	Debug	Serial data output for the JTAG test subsystem. High impedance except when scanning out data	Hi-Z	O	20-9

Table 20-2. 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	20-9
$\overline{\text{TRST}}$	Test reset	Debug	Resets the TAP controller asynchronously	—	I	20-9
THERM[0:1]	Thermal resistor access	Test	These pins tie directly to an internal resistor whose value varies linearly with temperature.	—	I	20-9
TEST_SEL	Test select	Test	Factory test. Must be negated for normal operation	—	I	20-9
$\overline{\text{LSSD\_MODE}}$	Test	Test	Factory Test. Refer to the *MPC8540 Integrated Processor Hardware Specifications* for proper treatment.		I	20-9
L1_TSTCLK	Test	Test	Factory Test. Refer to the *MPC8540 Integrated Processor Hardware Specifications* for proper treatment.		I	20-9
L2_TSTCLK	Test	Test	Factory Test. Refer to the *MPC8540 Integrated Processor Hardware Specifications* for proper treatment.		I	20-9

1 Although these signals are normally bidirectional, when sourcing debug information they are output only.

20.2.2 Detailed Signal Descriptions

This section describes the details of the debug, watchpoint monitor, and JTAG test signals

20.2.2.1 Debug Signals—Details

Table 20-3 describes all signals associated with device debug modes.

Table 20-3. 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 20-3. 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 20-1. 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 20-26 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

20.2.2.2 Watchpoint Monitor Trigger Signals—Details

Table 20-4 shows detailed descriptions of the watchpoint monitor and trace buffer signals.

Table 20-4. 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 MPC8540 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 20-4. 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 18, “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

20.2.2.3 Test Signals—Details

Table 20-5 shows detailed descriptions of the JTAG test signals.

Table 20-5. 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 20-5. JTAG Test and Other Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
$\overline{\text{TRST}}$	I	JTAG test reset	
		State Meaning	Asserted—Causes asynchronous initialization of the internal JTAG TAP controller. Must be asserted during power-on reset in order to properly initialize the JTAG TAP and for normal operation of the MPC8540. An unterminated input appears as a high signal level to the test logic due to an internal pull-up resistor. Negated— Normal operation
		Timing	See IEEE 1149.1 standard for more details
$\overline{\text{LSSD\_MODE}}$	I	Used for factory test. Refer to the *MPC8540 Integrated Processor Hardware Specifications* for proper treatment.	
$\overline{\text{L1\_TSTCLK}}$	I	Used for factory test. Refer to the *MPC8540 Integrated Processor Hardware Specifications* for proper treatment.	
$\overline{\text{L2\_TSTCLK}}$	I	Used for factory test. Refer to the *MPC8540 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 *MPC8540 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	

20.3 Memory Map/Register Definition

Table 20-6 shows the memory-mapped debug and watchpoint registers of the MPC8540.

Table 20-6. 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	20.3.1.1/20-11
0xE_2004	WMCR1—Watchpoint monitor control register 1	R/W	0x0000_0000	20.3.1.1/20-11
0xE_200C	WMAR—Watchpoint monitor address register	R/W	0x0000_0000	20.3.1.2/20-13
0xE_2014	WMAMR—Watchpoint monitor address mask register	R/W	0x0000_0000	20.3.1.3/20-14
0xE_2018	WMTMR—Watchpoint monitor transaction mask register	R/W	0x0000_0000	20.3.1.4/20-14
0xE_201C	WMSR—Watchpoint monitor status register	R/W	0x0000_0000	20.3.1.5/20-16
Trace Buffer Registers				
0xE_2040	TBCR0—Trace buffer control register 0	R/W	0x0000_0000	20.3.2.1/20-16
0xE_2044	TBCR1—Trace buffer control register 1	R/W	0x0000_0000	20.3.2.1/20-16

Table 20-6. 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	20.3.2.2/20-19
0xE_2054	TBAMR—Trace buffer address mask register	R/W	0x0000_0000	20.3.2.3/20-19
0xE_2058	TBTMR—Trace buffer transaction mask register	R/W	0x0000_0000	20.3.2.4/20-20
0xE_205C	TBSR—Trace buffer status register	R/W	0x0000_0000	20.3.2.5/20-21
0xE_2060	TBACR—Trace buffer access control register	R/W	0x0000_0000	20.3.2.6/20-22
0xE_2064	TBADHR—Trace buffer access data high register	R/W	0x0000_0000	20.3.2.7/20-22
0xE_2068	TBADR—Trace buffer access data register	R/W	0x0000_0000	20.3.2.8/20-23
Context ID Registers				
0xE_20A0	PCIDR—Programmed context ID register	R/W	0x0000_0000	20.3.3.1/20-24
0xE_20A4	CCIDR—Current context ID register	R/W	0x0000_0000	20.3.3.2/20-24
Other Registers				
0xE_20B0	TOSR—Trigger output source register	R/W	0x0000_0000	20.3.4.1/20-25

20.3.1 Watchpoint Monitor Register Descriptions

The following sections describe the control registers for the watchpoint monitor facility.

20.3.1.1 Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1)

The watchpoint monitor control registers (WMCR0, WMCR1) shown in Figure 20-2 and Figure 20-3 control the specification of watchpoint monitor events.

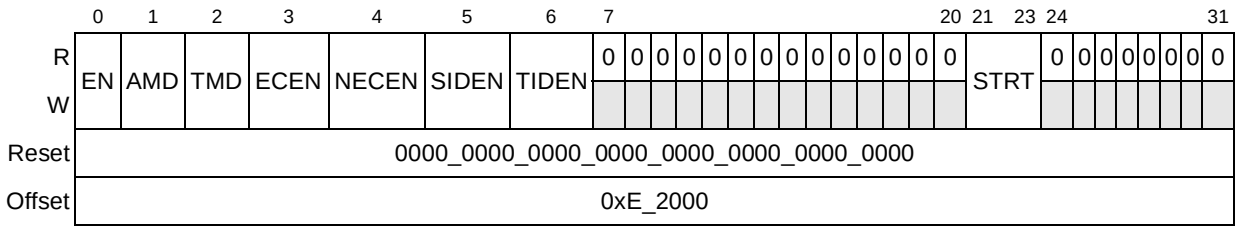


Figure 20-2. Watchpoint Monitor Control Register 0 (WMCR0)

Table 20-7 describes WMCR0 fields.

Table 20-7. 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 20.3.3, “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 20.3.3, “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 20-3 shows the WMCR1.

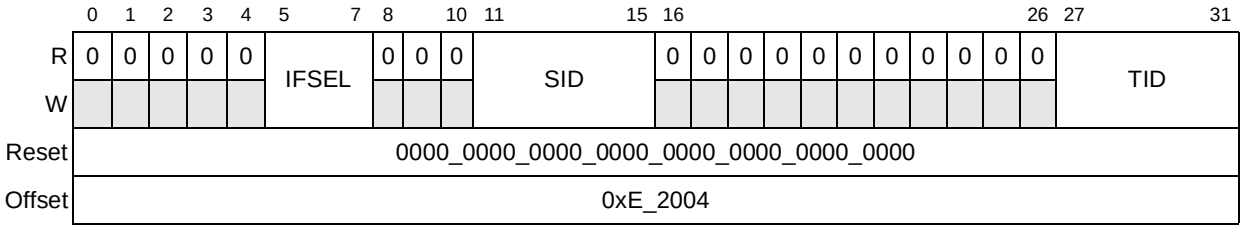


Figure 20-3. Watchpoint Monitor Control Register 1 (WMCR1)

Table 20-8 describes the WMCR1 fields.

Table 20-8. 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 20-26.
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 20-26.

20.3.1.2 Watchpoint Monitor Address Register (WMAR)

The watchpoint monitor address register (WMAR) shown in Figure 20-4 contains the address to match against if WMCR[AMD] is clear. Note that the transaction address is qualified with the bits described in Section 20.3.1.3, “Watchpoint Monitor Address Mask Register (WMAMR),” before being compared with WMAR. Note also that the contents of WMAR are not qualified with WMAMR.

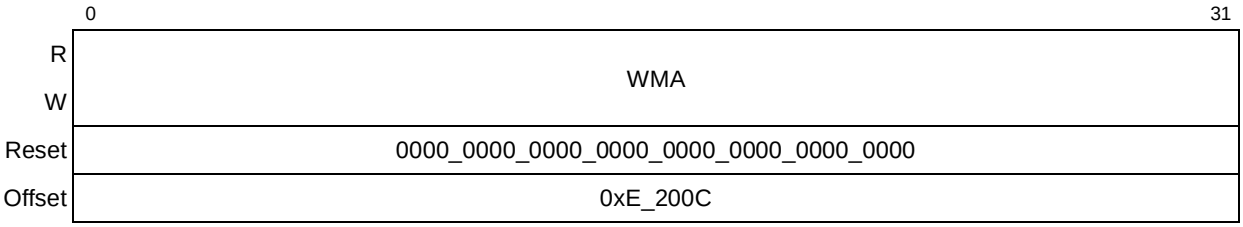


Figure 20-4. Watchpoint Monitor Address Register (WMAR)

Table 20-9 describes the WMAR fields.

Table 20-9. WMAR Field Descriptions

Bits	Name	Description
0–31	WMA	Watchpoint monitor address.

20.3.1.3 Watchpoint Monitor Address Mask Register (WMAMR)

The watchpoint monitor address mask register (WMAMR) shown in Figure 20-5 contains the mask that is applied to a transaction address before the address is compared with WMAR.

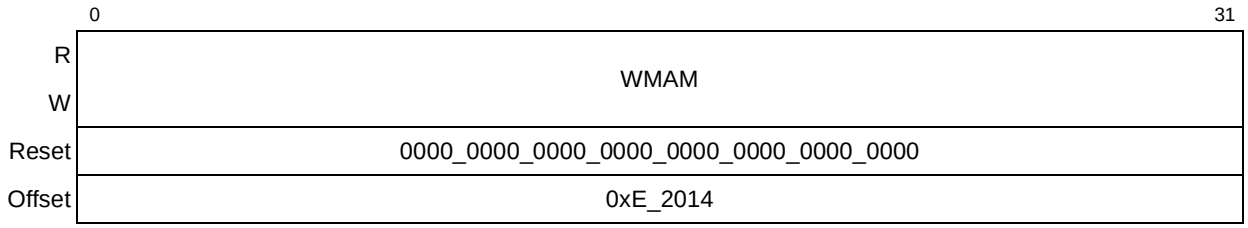


Figure 20-5. Watchpoint Monitor Address Mask Register (WMAMR)

Table 20-10 describes the WMAMR fields.

Table 20-10. 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.

20.3.1.4 Watchpoint Monitor Transaction Mask Register (WMTMR)

The watchpoint monitor transaction mask register (WMTMR), shown in Figure 20-6, specifies which transaction types to monitor. WMTMR allows users to qualify watchpoint events specifically with any combination of transaction types. As shown in Table 20-12, 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 20-12 lists transaction types associated with each WMTMR bit by interface.

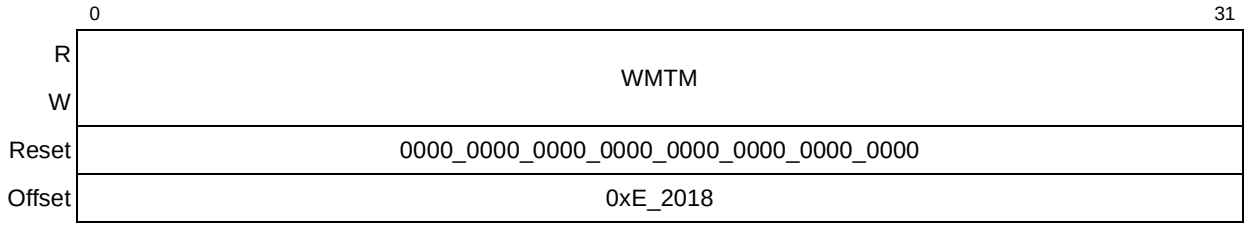


Figure 20-6. Watchpoint Monitor Transaction Mask Register (WMTMR)

Table 20-11 describes the WMTMR fields.

Table 20-11. WMTMR Field Descriptions

Bits	Name	Description
0–31	WMTM	Watchpoint monitor transaction mask. Each bit corresponds to a transaction type as defined in Table 20-12. 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 20-12. 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 20-12. 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

20.3.1.5 Watchpoint Monitor Status Register (WMSR)

The watchpoint monitor status register (WMSR) shown in Figure 20-7 indicates the state of the watchpoint monitor.

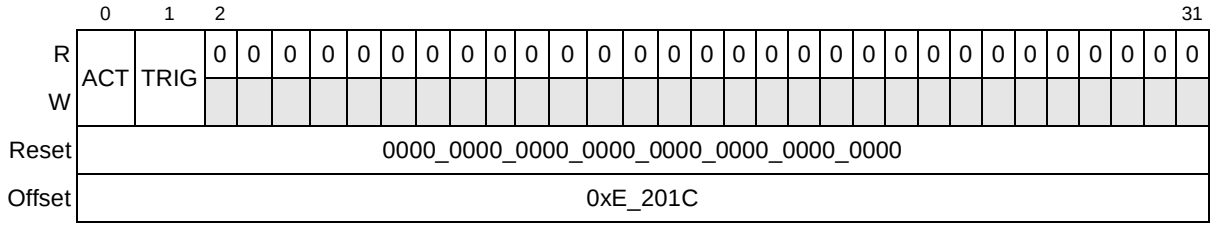


Figure 20-7. Watchpoint Monitor Status Register (WMSR)

Table 20-13 describes the WMSR fields.

Table 20-13. 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

20.3.2 Trace Buffer Register Descriptions

The following sections describes the trace buffer registers.

20.3.2.1 Trace Buffer Control Registers (TBCR0, TBCR1)

The trace buffer control registers (TBCR0, TBCR1), shown in Figure 20-8 and Figure 20-9, specify trace buffer events.

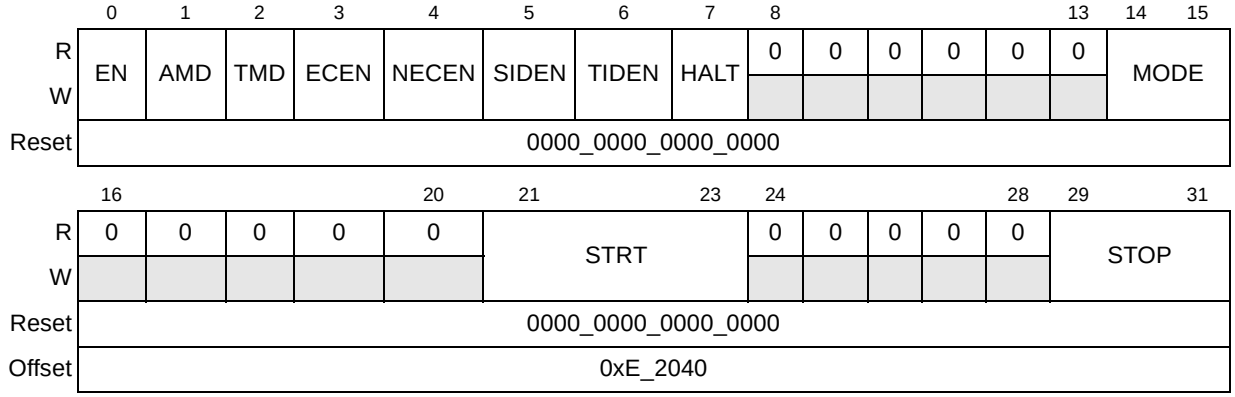


Figure 20-8. Trace Buffer Control Register 0 (TBCR0)

Table 20-14 describes the TBCR0 fields.

Table 20-14. 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 20.3.3, “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 20.3.3, “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 20-14. 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

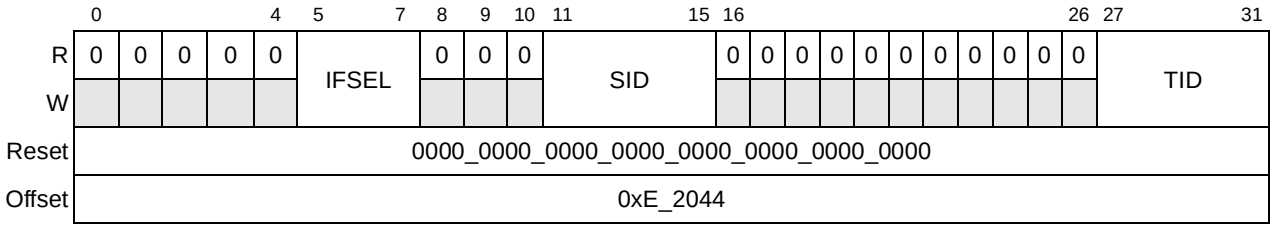


Figure 20-9. Trace Buffer Control Register 1 (TBCR1)

Table 20-15 describes the TBCR1 fields.

Table 20-15. 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 20-26.
16–26	—	Reserved
27–31	TID	Target ID. Specifies the target ID associated with TBCR0[TIDEN]. The target ID is defined in Table 20-26.

20.3.2.2 Trace Buffer Address Register (TBAR)

The trace buffer address register (TBAR) shown in Figure 20-10 contains the address to match against (if TBCR0[AMD] is zero). The transaction address is qualified with the bits described in Section 20.3.2.3, “Trace Buffer Address Mask Register (TBAMR),” before being compared with TBAR. Note that the contents of TBAR are not qualified with TBAMR.

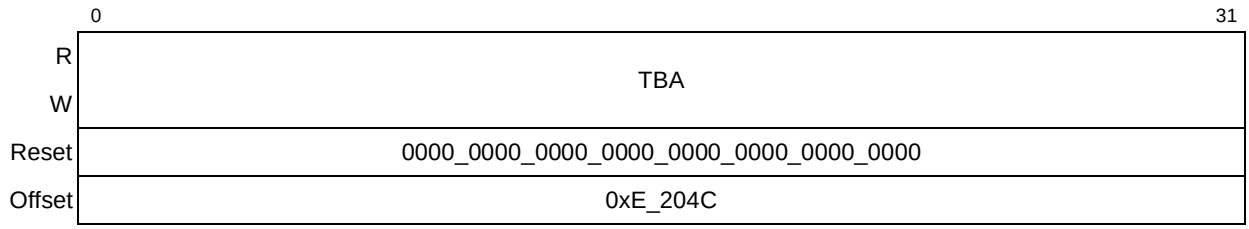


Figure 20-10. Trace Buffer Address Register (TBAR)

Table 20-16 describes the TBAR field.

Table 20-16. TBAR Field Descriptions

Bits	Name	Description
0–31	TBA	Trace buffer address

20.3.2.3 Trace Buffer Address Mask Register (TBAMR)

The trace buffer address mask register (TBAMR) shown in Figure 20-11 contains the mask that is applied to a transaction address before the address is compared with TBAR.

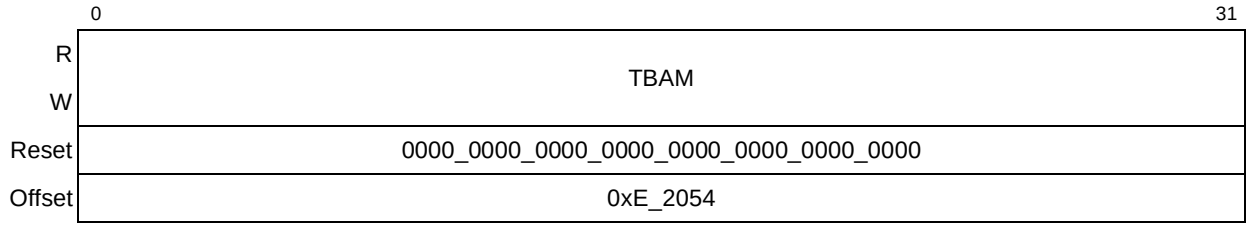


Figure 20-11. Trace Buffer Address Mask Register (TBAMR)

Table 20-17 describes the TBAMR field.

Table 20-17. 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.

20.3.2.4 Trace Buffer Transaction Mask Register (TBTMR)

The trace buffer transaction mask register (TBTMR) shown in Figure 20-12 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 20-12 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.

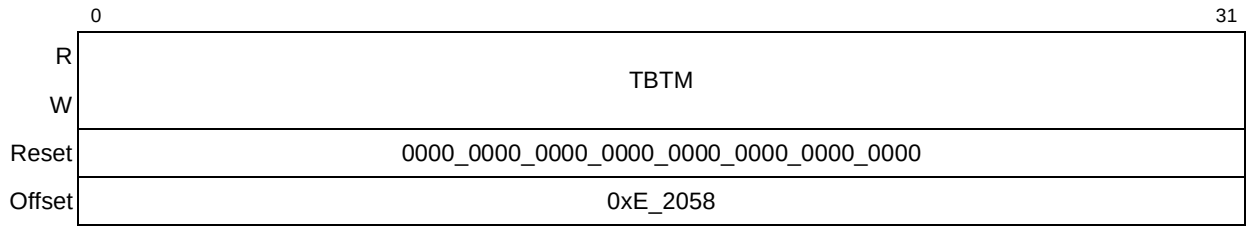


Figure 20-12. Trace Buffer Transaction Mask Register (TBTMR)

Table 20-18 describes the TBTMR field.

Table 20-18. TBTMR Field Descriptions

Bits	Name	Description
0–31	TBTM	Trace buffer transaction mask. Each bit corresponds to a transaction type as defined in Table 20-12. 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.

20.3.2.5 Trace Buffer Status Register (TBSR)

The trace buffer status register (TBSR) shown in Figure 20-13 indicates the operational state of the trace buffer.

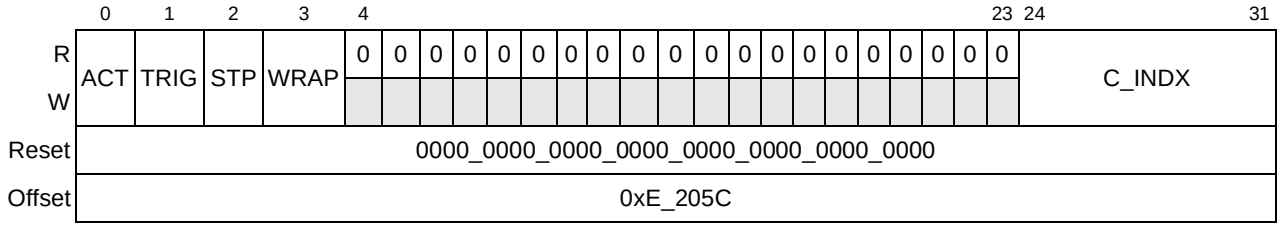


Figure 20-13. Trace Buffer Status Register (TBSR)

Table 20-19 describes the TBSR fields.

Table 20-19. 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.

20.3.2.6 Trace Buffer Access Control Register (TBACR)

The trace buffer access control register (TBACR), shown in Figure 20-14, 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.

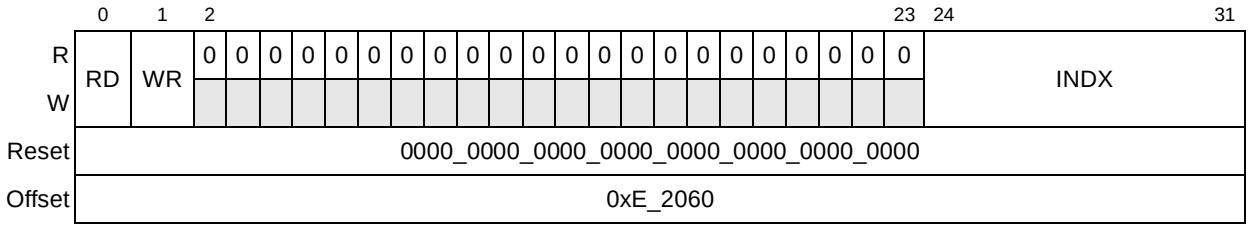


Figure 20-14. Trace Buffer Access Control Register (TBACR)

Table 20-20 describes the TBACR fields.

Table 20-20. 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]

20.3.2.7 Trace Buffer Access Data High Register (TBADHR)

The trace buffer access data high register (TBADHR), shown in Figure 20-15, 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.

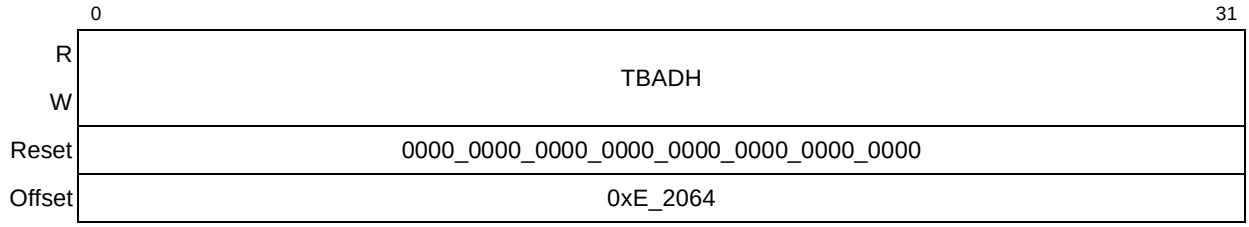


Figure 20-15. Trace Buffer Read High Register (TBADHR)

Table 20-21 describes TBADHR.

Table 20-21. 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.

20.3.2.8 Trace Buffer Access Data Register (TBADR)

The trace buffer access data register (TBADR), shown in Figure 20-16, 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.

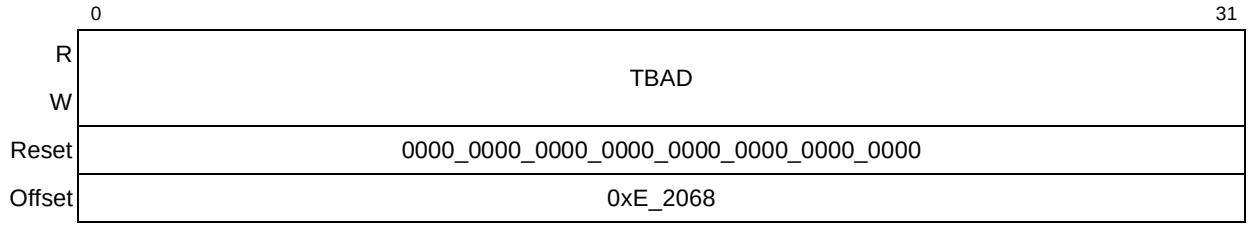


Figure 20-16. Trace Buffer Access Data Register (TBADR)

Table 20-22 describes the TBADR field.

Table 20-22. 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.

20.3.3 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.

20.3.3.1 Programmed Context ID Register (PCIDR)

The programmed context ID register (PCIDR), shown in Figure 20-17, 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 20.3.1.1, "Watchpoint Monitor Control Registers 0–1 (WMCR0, WMCR1)," for more information.

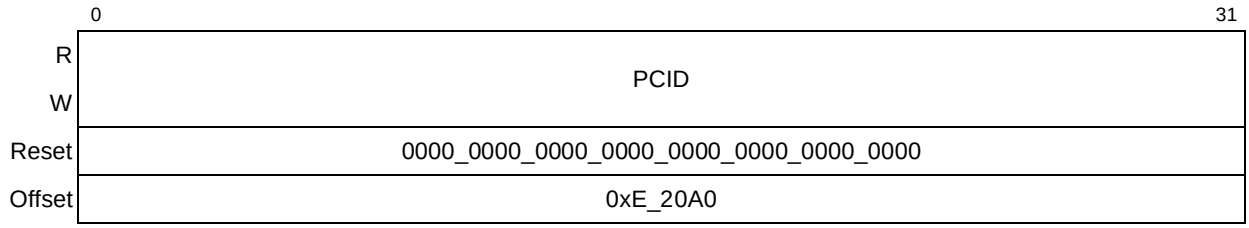


Figure 20-17. Programmed Context ID Register (PCIDR)

Table 20-23 describes the PCIDR field.

Table 20-23. 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.

20.3.3.2 Current Context ID Register (CCIDR)

The current context ID register (CCIDR) shown in Figure 20-18 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).

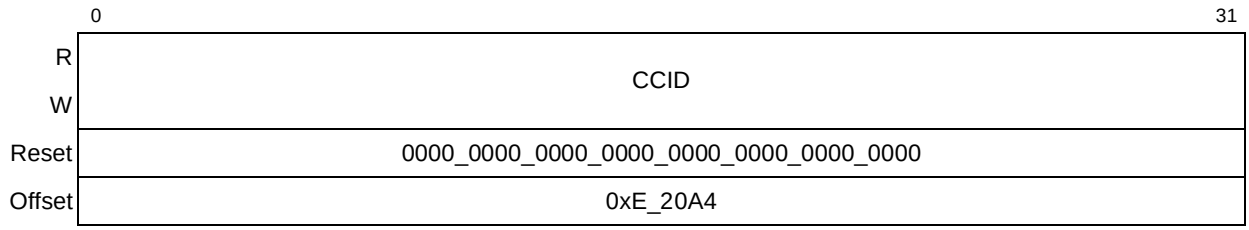


Figure 20-18. Current Context ID Register (CCIDR)

Table 20-24 describes the CCIDR field.

Table 20-24. 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.

20.3.4 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].

20.3.4.1 Trigger Out Source Register (TOSR)

The trigger out source register (TOSR) shown in Figure 20-19 specifies the source for TRIG_OUT. The three event-trigger sources are the following:

- The watchpoint monitor
- The trace buffer
- The performance monitor

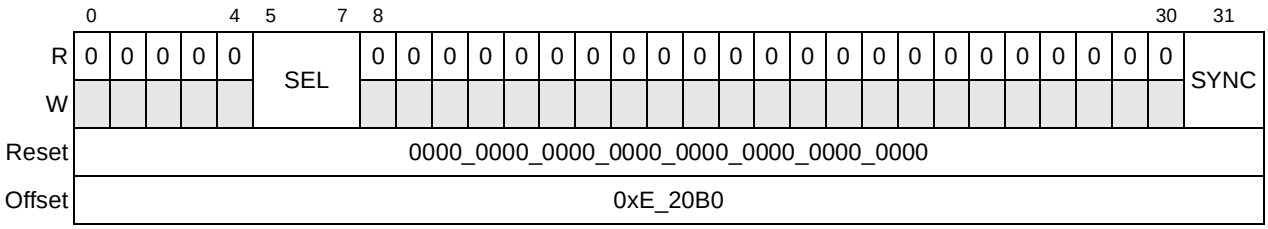


Figure 20-19. Trigger Out Source Register (TOSR)

Table 20-25 describes the TOSR fields.

Table 20-25. 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 18, "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

20.4 Functional Description

The debug features on the MPC8540 use the LBC interfaces, DDR SDRAM interface, and the PCI interface.

20.4.1 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 20-26 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 20-26. 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 20-26. 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	Reserved
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	FEC
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)

20.4.2 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.

20.4.3 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.

20.4.3.1 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 18.4.1.5, "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.

20.4.3.2 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 20-1. 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.

20.4.4 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 13.1.3.2, "Source ID Debug Mode."

20.4.5 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.

20.4.5.1 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 19-10.

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.

20.4.6 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.

20.4.6.1 Traced Data Formats (as a Function of TBCR1[IFSEL])

Figure 20-20 shows the trace buffer entry format for an ECM dispatch (CMD) transaction that is specified when TBCR1[IFSEL] = 000.

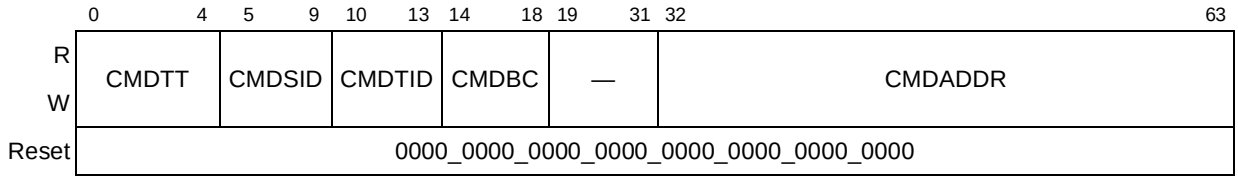


Figure 20-20. e500 Coherency Module Dispatch (CMD) Trace Buffer Entry

Table 20-27 describes the fields of CMD trace buffer entries.

Table 20-27. 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 20-12. 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 20-26. 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 20-26. 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 20-27. 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 20-21 shows the trace buffer entry format for the DDR SDRAM interface, TBCR1[IFSEL] = 001.

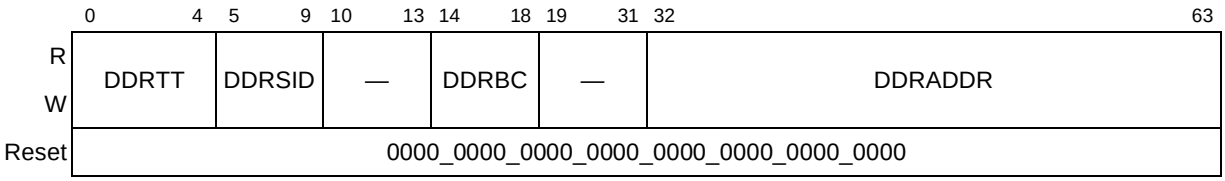


Figure 20-21. DDR Trace Buffer Entry

Table 20-28 describes the fields of DDR SDRAM trace buffer entries when TBCR1[IFSEL] = 001.

Table 20-28. 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 20-12. 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 20-26. 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 20-22 shows the PCI trace buffer entry format when TBCR1[IFSEL] = 010.

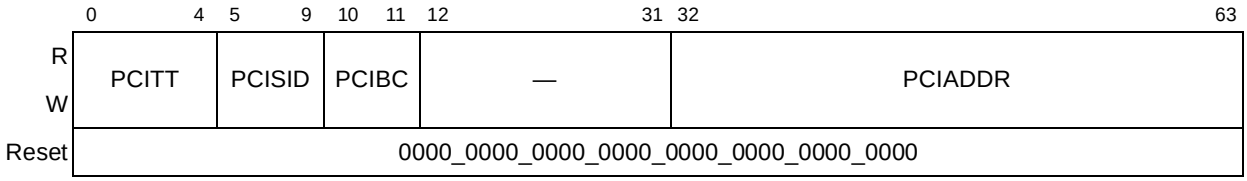


Figure 20-22. PCI Trace Buffer Entry

Table 20-29 describes the fields of PCI trace buffer entries when TBCR1[IFSEL] = 010.

Table 20-29. 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 20-12. 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 20-26. 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 20-23 shows the format for RapidIO interface trace buffer entries when TBCR1[IFSEL] = 011.

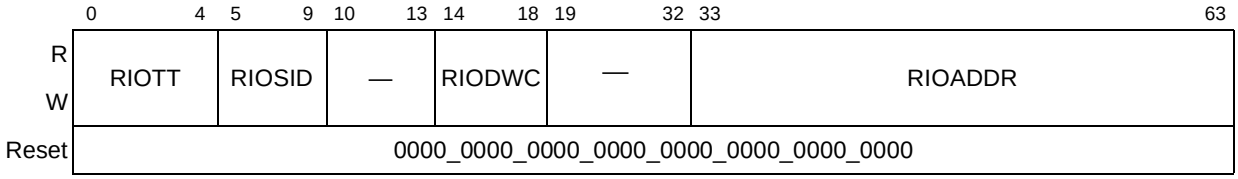


Figure 20-23. RapidIO Trace Buffer Entry

Table 20-30 describes the fields of the RapidIO interface trace buffer entries.

Table 20-30. 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 20-12. 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 20-26. For example, a value of 01100 identifies RapidIO as the transaction source.

Table 20-30. 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

20.5 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.

Chapter 21
10/100 Fast Ethernet Controller

The MPC8540 provides a 10/100 fast Ethernet controller (FEC) operating at 10 to 100 megabits per second (Mbps) that can be used as a device debug and maintenance port. The three-speed (10/100/1Gb) Ethernet controller (TSEC) is separate from this 10/100 Ethernet controller, and is described in Chapter 14, "Three-Speed Ethernet Controllers."

21.1 Introduction

The 10/100 Ethernet standard supports the IEEE 802.3 Ethernet frame format, backward compatibility for installed media, and the use of full- or half-duplex CSMA/CD.

The Ethernet protocol implements the bottom two layers of the open systems interconnection (OSI) 7-layer model, that is, the data link and physical layers. Figure 21-1 depicts the typical Ethernet protocol stack and the relationship to the OSI model.

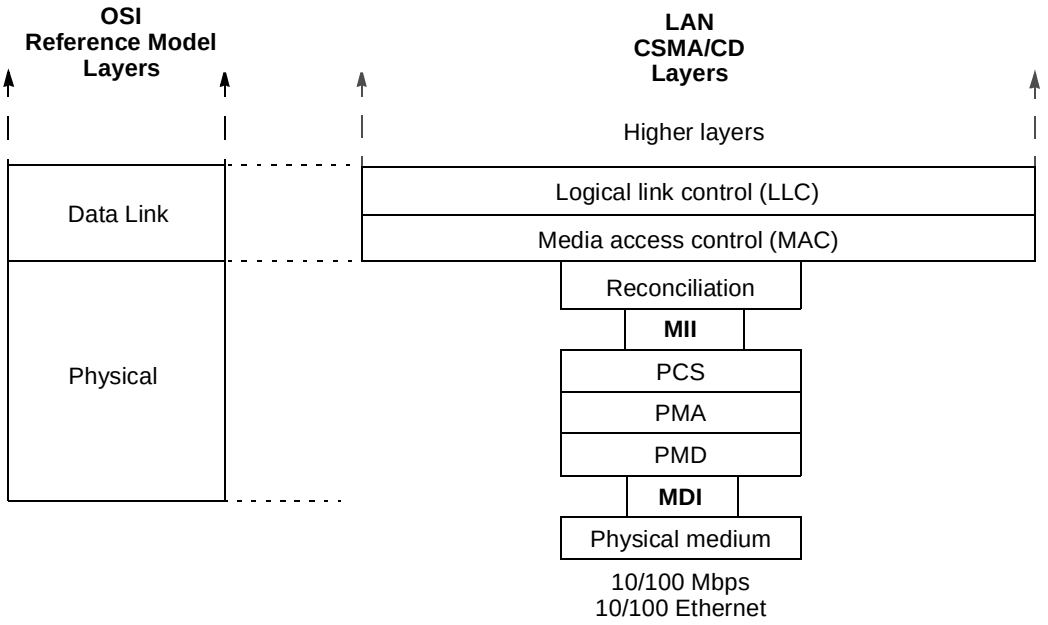


Figure 21-1. 10/100 Fast Ethernet Controller in Relation to the OSI Protocol Stack

Ethernet provides the following sublayers:

- Media access control (MAC) sublayer—The MAC sublayer provides a logical connection between the MAC and its peer station. Its primary responsibility is to initialize, control, and manage the connection with the peer station.
- Reconciliation sublayer—The reconciliation sublayer acts as a command translator. It maps the terminology and commands used in the MAC layer into electrical formats appropriate for the physical layer entities.
- MII (media-independent interface) sublayer—The MII sublayer provides a standard interface between the MAC layer and the physical layer for 10/100 Mbps 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.
- PMA (physical medium attachment) sublayer—The PMA sublayer is responsible for serializing code groups into a bit stream suitable for serial bit-oriented physical devices(SERDES) and vice versa. Synchronization is also performed for proper data decoding in this sublayer. The PMA sits between the PCS and the PMD sublayers.
- PMD (physical medium dependent) sublayer—The PMD sublayer is responsible for signal transmission. The typical PMD functionality includes amplifier, modulation, and wave shaping. Different PMD devices may support different media.
- MDI (medium-dependent interface) sublayer—MDI is a connector. It defines different connector types for different physical media and PMD devices.

Ethernet/IEEE 802.3 frames are based on the frame structure shown in Figure 21-2. The term “packet” is sometimes used to refer to the frame plus the preamble and start frame delimiter (SFD).

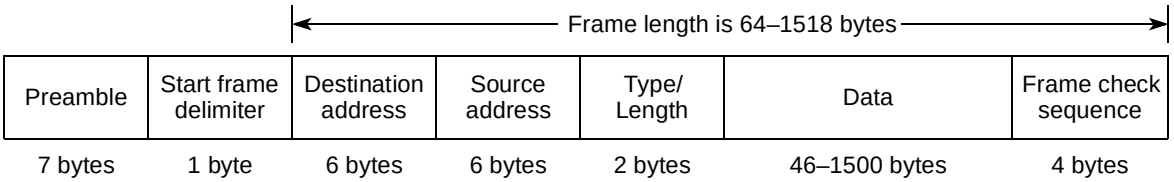


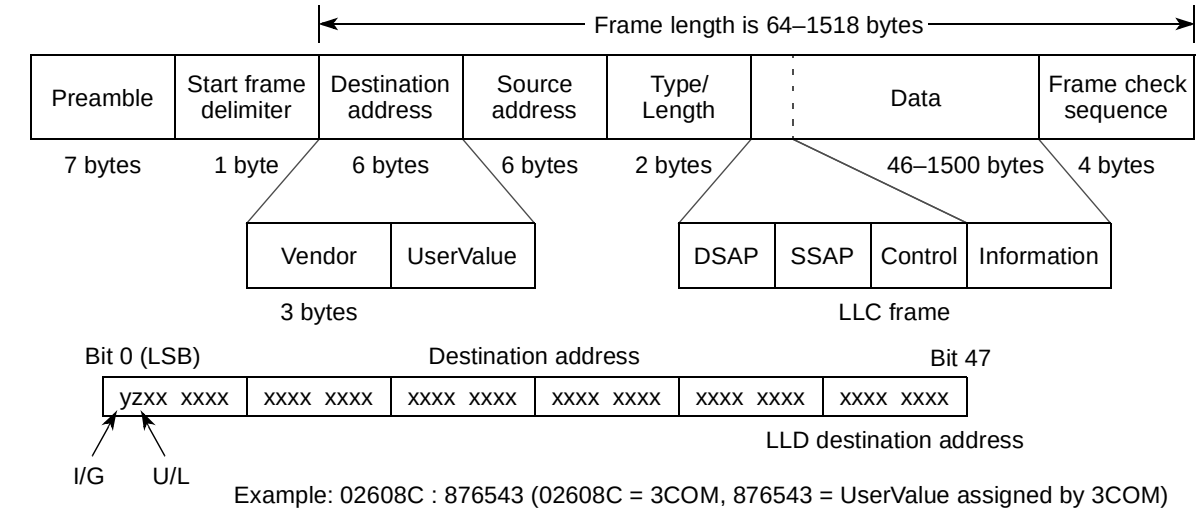
Figure 21-2. Ethernet/IEEE 802.3 Frame Structure

The elements of an Ethernet frame are as follows:

- Preamble—The 7-byte preamble of alternating ones and zeros used for receiver timing synchronization (each byte containing the value 0x55)
- Start frame delimiter (SFD)—A sequence of 0xD5 (10101011 because the bit ordering is lsb first) indicates the beginning of the frame

- 48-bit destination address (DA)—The first bit identifies the address as an individual address (0) or a group address (1). The second bit is used to indicate whether the address is locally-defined (1) or globally-defined (0)
- 48-bit source address (SA)—(Original versions of the IEEE 802.3 specification allowed 16-bit addressing, which has never been used widely.)
- Ethernet type field/IEEE 802.3 length field—The type field signifies the protocol (for example, TCP/IP) used in the rest of the frame. The length field specifies the length of the data portion of the frame. For both Ethernet and IEEE 802.3 frames to exist on the same LAN, the length field must be unique from any type fields used in Ethernet. This limitation requires that a type field be identified by a decimal number equal to or greater than 1536 (0x0600) but less than 65535 (0xFFFF). If the number, however, is between 0 and 1,500 (0x0000 through 0x05DC) then this field indicates the length of the MAC client data. The range from 1,501 to 1,535 (0x5DD through 0x5FF) was intentionally left undefined.
- Data and padding—Padding is optional. It is only needed if the data is smaller than 46 octets (one octet = one byte) to ensure the minimum frame size of 64 octets as specified in the IEEE 802.3 standard. In 802.3x the first two octets of the data field are used as opcode (OP) (pause = 0x0001) and the second two octets are used to transmit a pause time (PT) parameter (pausetime = 0x0000 for on and 0xFFFF for off). In addition, a third two-octet field can be used for an extended pause control parameter (PTE). Because the use of these fields varies with the protocol used, the ability to examine them and report their content can significantly accelerate Ethernet frame processing.
- Frame-check sequence (FCS)—Specifies the standard 32-bit cyclic redundancy check (CRC) obtained using the standard CCITT-CRC polynomial on all fields except the preamble, SFD and CRC.

Figure 21-3 details the Ethernet/IEEE 802.3 frame structure.



Note: The lsb of each octet is transmitted first.

Figure 21-3. Ethernet/IEEE 802.3 Frame Structure With More Details

Relative to Figure 21-3, the IEEE 802.3 section 3.11 (MAC frame format) defines the frame format such that the octets of a frame are transmitted from left to right (preamble first, the FCS last), and the bits of each octet are transmitted least-significant bit (lsb) first. The destination address example shown in Figure 21-3 (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 is 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 nonzero.

The FEC allows the flexibility to accelerate the identification and retrieval of all the standard and non-standard protocols mentioned above. Figure 21-4 depicts the block diagram of the FEC.

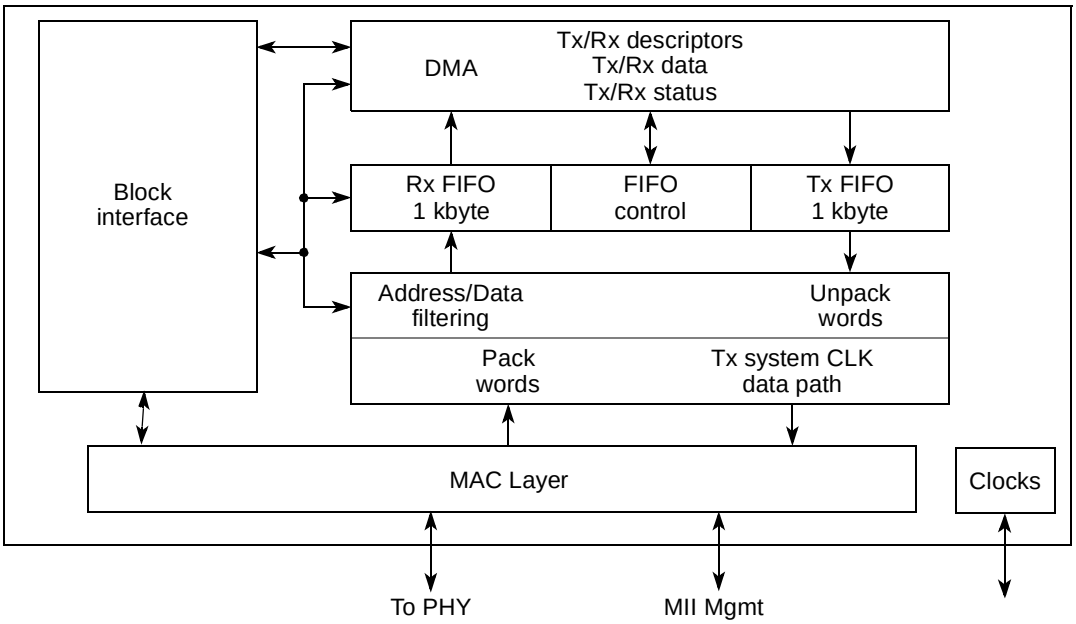


Figure 21-4. FEC Block Diagram

21.1.1 10/100 Fast Ethernet Controller Overview

The FEC is designed to support 10/100 Mbps Ethernet/802.3 networks and contains the following components:

- Ethernet media access controller (MAC)
- First-in first-out (FIFO) controller
- Direct memory access (DMA) controller

The most-significant byte of data in a receive or transmit data buffer corresponds to the most-significant byte of a frame, respectively.

The complete FEC is designed for single MAC applications. The FEC supports the 10/100 Mbps MII interface to connect to an external Ethernet transceiver.

The FEC software programming model is similar to the MPC8260 (PowerQUICC II) device. Hence, it enables Freescale Semiconductor customers to leverage already implemented Ethernet drivers, reducing the software development cycle.

21.2 Features

The MPC8540 FEC includes these distinctive features:

- IEEE 802.3, 802.3u, 802.3x, 802.3ac compliant
- Support for different Ethernet physical interfaces:
  - 10/100 Mbps IEEE 802.3 MII
  - 10 Mbps IEEE 802.3 MII
- Full- and half-duplex support
- IEEE 802.3 full-duplex flow control (automatic PAUSE frame generation or software programmed PAUSE frame generation and recognition)
- Support 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 processor’s L2 cache to reduce access latency

21.3 Modes of Operation

The FEC's primary operational modes are the following:

- Full- and half-duplex operation. Determined by MACCFG2 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 Mbps operation; MACCFG2 register's Full_Duplex bit is set), the MAC supports flow control. If flow control is enabled, it allows the MAC to receive or send PAUSE frames.
- 10 Mbps and 100 Mbps MII interface operation. The MAC-PHY interface operates in MII mode by configuring bits 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.

  Operation speed is determined by the FEC_TX_CLK and FEC_RX_CLK signals which are driven by the transceiver. The transceiver auto negotiates the speed.
- 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 21.5.3.7.1, "Individual Address Registers 0–7 (IADDRn)," Section 21.5.3.7.2, "Group Address Registers 0–7 (GADDRn)," Section 21.5.3.4.1, "Receive Control Register (RCTRL)," and Section 21.6.2.5, "Frame Recognition."
- Internal loop back

  Internal loop back mode is selected via the Loop_Back bit in the MACCFG1 register. See Section 21.6.2.9, "Internal and External Loop Back," for details.

21.4 External Signal Descriptions

This section defines the FEC-to-chip signals. The buses are described using the bus convention used in IEEE 802.3 because the PHY follows this same convention (that is, TxD[3:0] means 0 is the lsb). Notice that except for external physical interfaces the buses and registers follow a big-endian format.

The FEC network interface supports the MII option that requires 15 I/O signals. The MII option supports both 10- and 100-Mbps Ethernet rates.

21.4.1 Detailed Signal Descriptions

Table 21-1 provides a detailed description of the FEC interface signals. The modes follow the IEEE 802.3 standard, 2000 Edition. Input signals not used are internally disabled. Output signals not used are driven low.

Table 21-1. FEC Signals—Detailed Signal Descriptions

Signal	I/O	Description	
FEC_COL	I	Collision. The behavior of this signal is not specified when in full-duplex mode.	
		State Meaning	Asserted/Negated—Asserted upon detection of a collision. It must remain asserted while the collision persists.
		Timing	Asserted/Negated—Not required to transition synchronously with FEC_TX_CLK or FEC_RX_CLK.
FEC_CRS	I	Carrier sense. Used as SDET (signal detect), an optional signal that some PHYs generate.	
		State Meaning	Asserted/Negated—Asserted when the transmit or receive medium is not idle. In case of a collision, FEC_CRS must stay asserted through the duration of the collision.
		Timing	Asserted/Negated—Not required to transition synchronously with FEC_TX_CLK or FEC_RX_CLK.
FEC_RX_CLK	I	Receive clock. This is a continuous clock (2.5, or 25 MHz) that provides a timing reference for FEC_RX_DV, FEC_RXD, and FEC_RX_ER.	
FEC_RX_DV	I	Receive data valid. If FEC_RX_DV is asserted, the PHY is indicating that valid data is present on the MII interfaces.	
FEC_RXD[3:0]	I	Receive data in. Represents a nibble of data to be transferred from the PHY to the MAC when FEC_RX_DV is asserted. A completely formed SFD must be passed across the MII. When FEC_RX_DV is not asserted, FEC_RXD has no meaning.	
FEC_RX_ER	I	Receive error	
		State Meaning	Asserted—When FEC_RX_ER and FEC_RX_DV are asserted, the PHY has detected an error in the current frame. Negated—The PHY has not detected an error in the current frame.
FEC_TX_CLK	I	Transmit clock in. FEC_TX_CLK is a continuous clock (with a frequency of 2.5 or 25 MHz) that provides a timing reference for FEC_TX_EN, FEC_TXD, and FEC_TX_ER. FEC_TX_CLK is generated in the PHY and provided to the MAC.	
FEC_TXD[3:0]	O	Transmit data out	
		State Meaning	Asserted/Negated—Represents a nibble of data to be sent from the MAC to the PHY when FEC_TX_EN is asserted; has no meaning if FEC_TX_EN is negated.
FEC_TX_EN	O	Transmit data valid	
		State Meaning	Asserted—The MAC is indicating that valid data is present on the MII. Negated—Valid data is not guaranteed to be present on the MII.
		Timing	Asserted/Negated—This signal transitions synchronously with respect to FEC_TX_CLK.

Table 21-1. FEC Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
FEC_TX_ER	O	Transmit error	
		State Meaning	Asserted/Negated—Assertion of this signal for one or more clock cycles while FEC_TX_EN is asserted causes the PHY to transmit one or more illegal symbols. Asserting FEC_TX_ER has no effect when operating at 10 Mbps or when FEC_TX_EN is negated.
		Timing	Asserted/Negated—This signal transitions synchronously with respect to FEC_TX_CLK.

21.5 Memory Map/Register Definition

The FEC uses a software model similar to that employed by the Fast Ethernet function supported on the Freescale Semiconductor MPC8260 CPM FCC and in the 10/100 controller of the MPC860T.

The FEC is programmed by a combination of control/status registers (CSRs) 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 descriptors are described in Section 21.6.3, “Buffer Descriptors.”

The MII registers are described in this section.

21.5.1 Top-Level Module Memory Map

The FEC implementation requires 4 Kbytes of memory-mapped space, of which more than 1 Kbyte is reserved for future expansion. The space is divided into the following sections:

- General control/status registers
- Transmit-specific control/status registers
- Receive specific control/status registers
- MAC registers
- Hash function registers

Table 21-2 defines the top-level memory map.

Table 21-2. Module Memory Map Summary

Address	Function
000–0FF	FEC general control/status registers
100–2FF	FEC transmit control/status registers
300–4FF	FEC receive control/status registers
500–5FF	FEC MAC registers
600–7FF	Reserved
800–8FF	FEC HASH function registers
900–AFF	Reserved
B00–BFF	FEC attribute registers
C00–FFF	Future expansion space

21.5.2 Detailed Memory Map

Table 21-3 shows the address of each register, the name of the register, and a cross-reference to the complete description of the register.

Table 21-3. Module Memory Map

Offset	Name	Access [1]	Reset	Section/Page
FEC General Control and Status Registers				
0x2_6000–0x2_600C	Reserved	R	0x0000_0000	—
0x2_6010	IEVENT—Interrupt event register	R	0x0000_0000	21.5.3.1.1/21-13
0x2_6014	IMASK—Interrupt mask register	R/W	0x0000_0000	21.5.3.1.2/21-16
0x2_6018	EDIS—Error disabled register	R/W	0x0000_0000	21.5.3.1.3/21-18
0x2_601C–0x2_6020	Reserved	R	0x0000_0000	—
0x2_6024	MINFLR—Minimum frame length register	R/W	0x0000_0040	21.5.3.1.4/21-18
0x2_6028	PTV—Pause time value register	R/W	0x0000_0000	21.5.3.1.5/21-19
0x2_602C	DMACTRL—DMA control register	R/W	0x0000_0000	21.5.3.1.6/21-20
0x2_6030–0x2_6088	Reserved	R	0x0000_0000	—
FEC FIFO Control and Status Registers				
0x2_604C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	21.5.3.2.1/21-22
0x2_608C	FIFO_TX_THR—FIFO transmit threshold register	R/W	0x0000_0080	21.5.3.2.2/21-23

Table 21-3. Module Memory Map (continued)

Offset	Name	Access [1]	Reset	Section/Page
0x2_6090–0x2_6094	Reserved	R	0x0000_0000	—
0x2_6098	FIFO_TX_STARVE—FIFO transmit starve register	R/W	0x0000_0020	21.5.3.3.4/21-27
0x2_609C	FIFO_TX_STARVE_SHUTOFF—FIFO transmit starve shutoff register	R/W	0x0000_0080	21.5.3.3.5/21-28
0x2_60A0–0x2_60FC	Reserved	R	0x0000_0000	—
FEC Transmit Control and Status Registers				
0x2_6100	TCTRL—Transmit control register	R/W	0x0000_0000	21.5.3.3.1/21-25
0x2_6104	TSTAT—Transmit status register	R/W	0x0000_0000	21.5.3.3.2/21-26
0x2_6108	Reserved	R	0x0000_0000	—
0x2_610C	TBDLEN—TxBD data length	R	0x0000_0000	21.5.3.3.3/21-27
0x2_6110–0x2_6120	Reserved	R	0x0000_0000	—
0x2_6124	CTBPTR—Current TxBD pointer register	R	0x0000_0000	21.5.3.3.4/21-27
0x2_6128–0x2_6180	Reserved	R	0x0000_0000	—
0x2_6184	TBPTR—TxBD pointer register	R/W	0x0000_0000	21.5.3.3.5/21-28
0x2_6188–0x2_6200	Reserved	R	0x0000_0000	—
0x2_6204	TBASE—TxBD base address register	R/W	0x0000_0000	21.5.3.3.6/21-28
0x2_6208–0x2_62AC	Reserved	R	0x0000_0000	—
0x2_62B0	OSTBD—Out-of-sequence TxBD register	R/W	0x0000_0000	21.5.3.3.7/21-29
0x2_62B4	OSTBDP—Out-of-sequence TxBD pointer register	R/W	0x0000_0000	21.5.3.3.8/21-31
0x2_62B8–0x2_62FC	Reserved	R	0x0000_0000	—
FEC Receive Control and Status Registers				
0x2_6300	RCTRL—Receive control register	R/W	0x0000_0000	21.5.3.4.1/21-31
0x2_6304	RSTAT—Receive status register	R/W	0x0000_0000	21.5.3.4.2/21-32
0x2_6308	Reserved	R	0x0000_0000	—
0x2_630C	RBDLEN—RxBD data length register	R	0x0000_0000	21.5.3.4.3/21-33
0x2_6310–0x2_6320	Reserved	R	0x0000_0000	—
0x2_6324	CRBPTR—Current RxBD pointer register	R	0x0000_0000	21.5.3.4.4/21-33

Table 21-3. Module Memory Map (continued)

Offset	Name	Access [1]	Reset	Section/Page
0x2_6328–0x2_633C	Reserved	R	0x0000_0000	—
0x2_6340	MRBLR—Maximum receive buffer length register	R/W	0x0000_0000	21.5.3.4.5/21-34
0x2_6344–0x2_6380	Reserved	R	0x0000_0000	—
0x2_6384	RBPTR—RxBD pointer register	R/W	0x0000_0000	21.5.3.4.6/21-35
0x2_6388–0x2_6400	Reserved	R	0x0000_0000	—
0x2_6404	RBASE—RxBD base address register	R/W	0x0000_0000	21.5.3.4.7/21-35
0x2_6408–0x2_64FC	Reserved	R	0x0000_0000	—
FEC MAC Registers				
0x2_6500	MACCFG1—MAC configuration register 1	R/W	0x0000_0000	21.5.3.6.1/21-38
0x2_6504	MACCFG2—MAC configuration register 2	R/W	0x0000_7000	21.5.3.6.2/21-40
0x2_6508	IPGIFG—Inter-packet gap/inter-frame gap register	R/W	0x4060_5060	21.5.3.6.3/21-41
0x2_650C	HAFDUP—Half-duplex register	R/W	0x00A0_F037	21.5.3.6.4/21-41
0x2_6510	MAXFRM—Maximum frame length register	R/W	0x0000_0600	21.5.3.6.5/21-42
0x2_6514–0x2_6538	Reserved	R	0x0000_0000	—
0x2_653C	IFSTAT—Interface status register	R	0x0000_0000	21.5.3.6.6/21-43
0x2_6540	MACSTNADDR1—Station address register, part 1	R/W	0x0000_0000	21.5.3.6.7/21-43
0x2_6544	MACSTNADDR2—Station address register, part 2	R/W	0x0000_0000	21.5.3.6.8/21-44
0x2_6548–0x2_67FC	Reserved	R	0x0000_0000	—
FEC Hash Function Registers				
0x2_6800	IADDR0—Individual address register 0	R/W	0x0000_0000	21.5.3.7.1/21-45
0x2_6804	IADDR1—Individual address register 1	R/W	0x0000_0000	
0x2_6808	IADDR2—Individual address register 2	R/W	0x0000_0000	
0x2_680C	IADDR3—Individual address register 3	R/W	0x0000_0000	
0x2_6810	IADDR4—Individual address register 4	R/W	0x0000_0000	
0x2_6814	IADDR5—Individual address register 5	R/W	0x0000_0000	
0x2_6818	IADDR6—Individual address register 6	R/W	0x0000_0000	
0x2_681C	IADDR7—Individual address register 7	R/W	0x0000_0000	
0x2_6820–0x2_687C	Reserved	R	0x0000_0000	—

Table 21-3. Module Memory Map (continued)

Offset	Name	Access [1]	Reset	Section/Page
0x2_6880	GADDR0—Group address register 0	R/W	0x0000_0000	21.5.3.7.2/21-46
0x2_6884	GADDR1—Group address register 1	R/W	0x0000_0000	
0x2_6888	GADDR2—Group address register 2	R/W	0x0000_0000	
0x2_688C	GADDR3—Group address register 3	R/W	0x0000_0000	
0x2_6890	GADDR4—Group address register 4	R/W	0x0000_0000	
0x2_6894	GADDR5—Group address register 5	R/W	0x0000_0000	
0x2_6898	GADDR6—Group address register 6	R/W	0x0000_0000	
0x2_689C	GADDR7—Group address register 7	R/W	0x0000_0000	
0x2_68A0–0x2_6AFF	Reserved	R	0x0000_0000	—
FEC Attribute Registers				
0x2_6B00–0x2_6BF4	Reserved	R	0x0000_0000	—
0x2_6BF8	ATTR—Attribute register	R/W	0x0000_0000	21.5.3.8.1/21-46
0x2_6BFC	ATTRELI—Attribute EL & EI register	R/W	0x0000_0000	21.5.3.8.2/21-47
FEC Future Expansion Space				
0x2_6C00–0x2_6FFF	Reserved	R	0x0000_0000	—

[1] R = Read Only, W = Write Only, R/W = Read and Write, LH = Latches High, SC = Self-clearing.

21.5.3 Memory-Mapped Register Descriptions

This section provides a detailed description of the FEC registers. Because all of these registers are 32 bits wide, only 32-bit register accesses are supported.

21.5.3.1 FEC General Control and Status Registers

This section describes general control and status registers used for both transmitting and receiving Ethernet frames. All of the registers are 32 bits wide.

21.5.3.1.1 Interrupt Event Register (IEVENT)

If an event occurs that sets a bit in the interrupt event (IEVENT) register, shown in Figure 21-5, an interrupt is generated if the corresponding bit in the interrupt enable register (IMASK) is also set. The bit in the interrupt event register is cleared if a 1 is written to that bit position. A write of 0 has no effect.

These interrupts can be divided into operational interrupts, transceiver/network error interrupts, and internal error interrupts. Interrupts that may occur in normal operation are:

- GTSC, GRSC, TXF, TXB, TXC, RXF, RXB, RXC, and 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

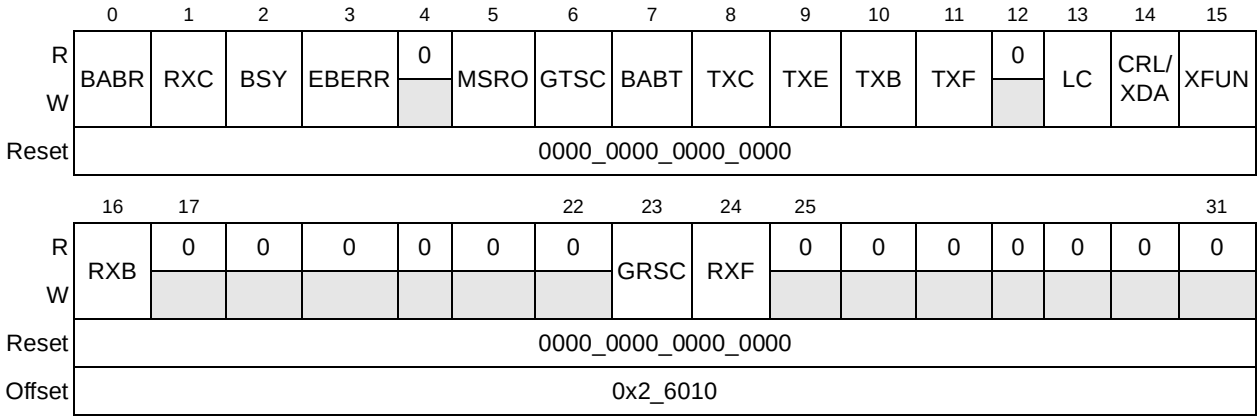


Figure 21-5. IEVENT Register Definition

Table 21-4 describes the fields of the IEVENT register.

Table 21-4. IEVENT Field Descriptions

Bits	Name	Description
0	BABR	Babbling receive error. This bit indicates that a frame was received with length in excess of the MAC's maximum frame length register while MACCFG2[Huge Frame] is set. 0 Excessive frame not received 1 Excessive frame received
1	RXC	Receive control interrupt. A control frame was received. If MACCFG1[Rx_Flow] is set, a pause operation is performed lasting for the duration specified in the received pause control frame and beginning when the frame was received. 0 Control frame not received 1 Control frame received
2	BSY	Busy condition interrupt. Indicates that a frame was received and discarded due to a lack of buffers. When IEVENT[BSY] is set RSTAT[QHLT] is also set. In order to begin receiving packets again, the user must clear RSTAT[QHLT]. This bit and RSTAT[QHLT] are set whenever the FEC reads an RxBD with its Empty field cleared. 0 No frame received and discarded. 1 Frame received and discarded.

Table 21-4. IEVENT Field Descriptions (continued)

Bits	Name	Description
3	EBERR	Ethernet bus error. This bit indicates that a system bus error occurred while a DMA transaction was underway. If the EBERR is set while transmission is in progress, the DMA stops sending data to the Tx FIFO which eventually causes an underrun error (XFUN) 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 No graceful stop interrupt. 1 Graceful stop requested.
7	BABT	Babbling transmit error. This bit indicates that the transmitted frame length has exceeded the value in the MAC's Maximum Frame Length register and MACCFG2[Huge Frame] is cleared. Frame truncation occurs when this condition occurs. TxBD[TXTRUNC] is set in the last TxBD (TxBD[L] is set) of the frame. 0 Transmitted frame length not exceeding maximum frame length. 1 Transmitted frame length exceeding maximum frame length.
8	TXC	Transmit control interrupt. This bit indicates that a control frame was transmitted. 0 Control frame not transmitted. 1 Control frame transmitted.
9	TXE	Transmit error. Indicates that an error occurred on the transmitted channel that caused TSTAT[THLT] to be set by the FEC. This bit is set whenever any transmit error occurs that causes the transmitter to halt (EBERR, LC, CRL/XDA, XFUN). It is not set if DMACTRL[WOP] is set and the FEC runs out of TxBDs to process. In order to begin transmitting packets again, the user must clear TSTAT[THLT]. 0 No transmit channel error occurred. 1 Transmit channel error occurred.
10	TXB	Transmit buffer. 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. Indicates that a frame was transmitted and that the last corresponding transmit buffer descriptor (TxBD) was updated. This only occurs if the I (Interrupt) bit in the status word of the buffer descriptor is set. 0 No frame transmitted/TxBD not updated. 1 Frame transmitted/TxBD updated.
12	—	Reserved
13	LC	Late collision. This bit indicates that a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame is truncated with a bad CRC and the remainder of the frame is discarded. 0 No late collision occurred. 1 Late collision occurred.

Table 21-4. IEVENT Field Descriptions (continued)

Bits	Name	Description
14	CRL/ XDA	Collision retry limit/excessive defer abort. 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. 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. Indicates that a receive buffer descriptor was updated which had the I (Interrupt) bit set in its status word and was not the last buffer descriptor of the frame. 0 Receive buffer descriptor not updated. 1 Receiver buffer descriptor updated.
17–22	—	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.
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

21.5.3.1.2 Interrupt Mask Register (IMASK)

The interrupt mask register, shown in Figure 21-6, 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.

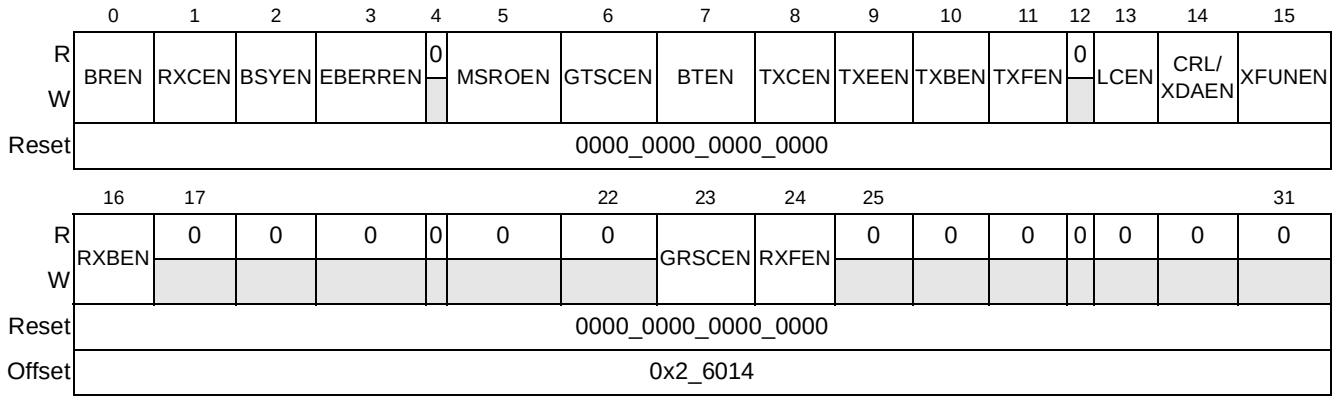


Figure 21-6. IMASK Register Definition

Table 21-5 describes the fields of the IMASK register.

Table 21-5. IMASK Field Descriptions

Bits	Name	Description
0	BREN	Babbling receiver interrupt enable
1	RXCEN	Receive control interrupt enable
2	BSYEN	Busy interrupt enable
3	EBERREN	Ethernet controller bus error enable
4	—	Reserved
5	MSROEN	MSTAT register overflow interrupt enable
6	GTSCEN	Graceful transmit stop complete interrupt enable
7	BTEN	Babbling transmitter interrupt enable
8	TXCEN	Transmit control interrupt enable
9	TXEEN	Transmit error interrupt enable
10	TXBEN	Transmit buffer interrupt enable
11	TXFEN	Transmit frame interrupt enable
12	—	Reserved
13	LCEN	Late collision enable
14	CRL/XDAEN	Collision retry limit / excessive defer abort enable
15	XFUNEN	Transmit FIFO underrun enable
16	RXBEN	Receive buffer interrupt enable
17–22	—	Reserved
23	GRSCEN	Graceful receive stop complete interrupt enable
24	RXFEN	Receive frame interrupt enable
25–31	—	Reserved

21.5.3.1.3 Error Disabled Register (EDIS)

The error disabled register allows the user to control the reporting of errors by the FEC. The IEVENT bit corresponding to an error is not set if the error is disabled in the EDIS register. Figure 21-7 shows the EDIS register.

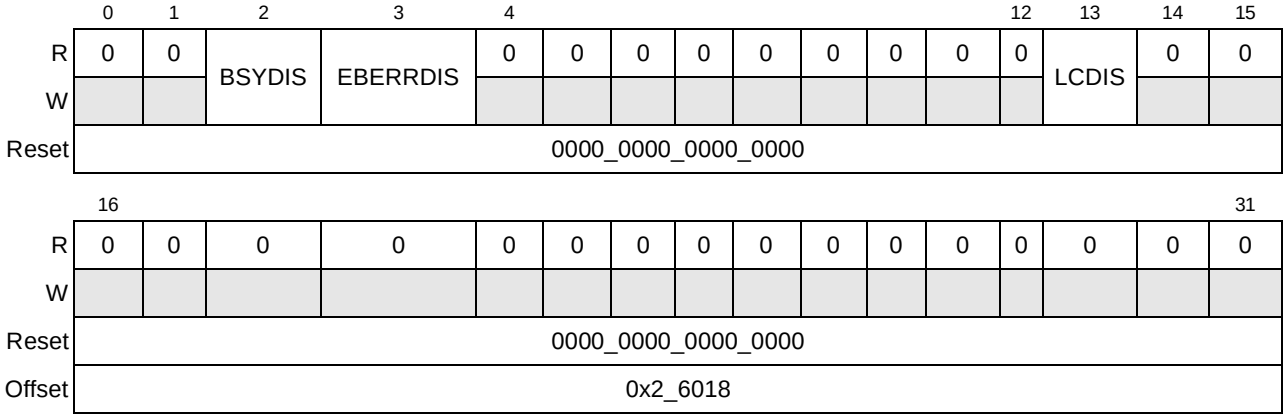


Figure 21-7. EDIS Register Definition

Table 21-6 describes the fields of the EDIS register.

Table 21-6. EDIS Field Descriptions

Bits	Name	Description
0–1	—	Reserved
2	BSYDIS	Busy disabled
3	EBERRDIS	Ethernet controller bus error disabled
4–12	—	Reserved
13	LCDIS	Late collision disabled
14–31	—	Reserved

21.5.3.1.4 Minimum Frame Length Register (MINFLR)

MINFLR, shown in Figure 21-8, is written by the user and tells the FEC the smallest packet to accept and place in a receive buffer pointed to by the RxBD.

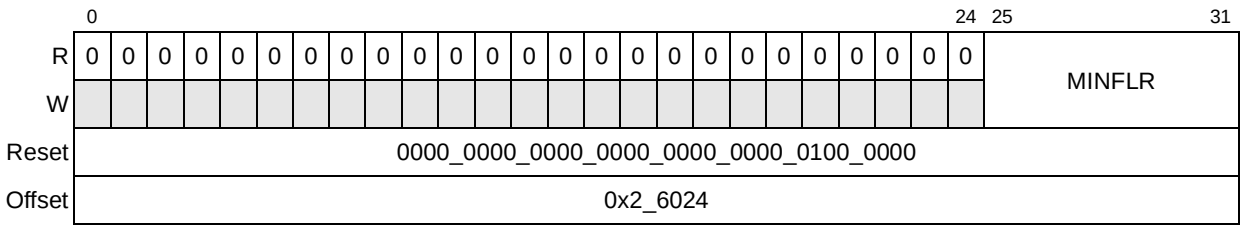


Figure 21-8. MINFLR Register Definition

Table 21-7 describes the fields of the MINFLR register.

Table 21-7. MINFLR Field Descriptions

Bits	Name	Description
0–24	—	Reserved
25–31	MINFLR	Minimum receive frame length (typically 64 decimal, which is the maximum). Ethernet discards incoming frames shorter than MINFLR unless RCTRL[RSF] (receive short frames) is set, in which case RxBD[SH] (frame too short) is set in the last RxBD. Unlike the MPC8260, in which pads are added to make the transmit frame equal to MINFLR bytes, if padding is requested, the FEC always pads transmit frames to 64 bytes ignoring MINFLR. MINFLR is only used to determine the minimum size of acceptable receive frames.

21.5.3.1.5 Pause Time Value Register (PTV)

PTV, shown in Figure 21-9, is written by the user to store the pause duration used when the FEC initiates a pause frame through TCTRL[TFC_PAUSE]. The low-order 16 bits (PT) represent the pause time and the high-order 16 bits (PTE) represent the extended pause control parameter. The pause time is measured in units of pause_quanta, equal to 512 bit times. The pause time can range from 0 to 65,535 pause_quanta, or 0 to 33,553,920 bit times. See Section 21.6.2.6, “Flow Control,” for additional details.

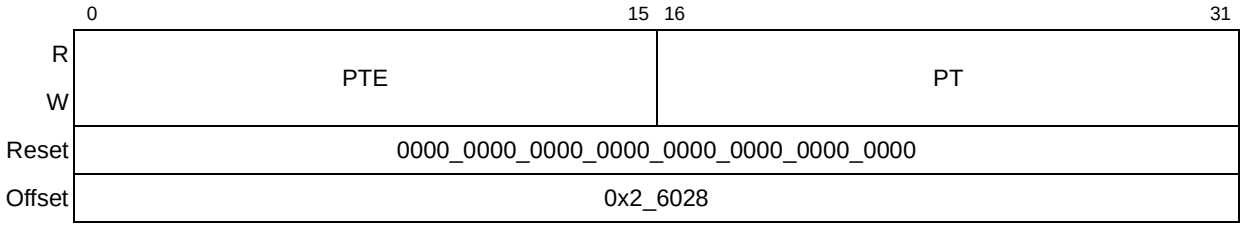


Figure 21-9. PTV Register Definition

Table 21-8 describes the fields of the PTV register.

Table 21-8. PTV Field Descriptions

Bits	Name	Description
0–15	PTE	Extended pause control. This field allows software to add a 16-bit additional control parameter into the pause frame to be sent when TCTRL[TFC_PAUSE] is set. Note that current IEEE 802.3 pause frame format requires this parameter to be set to 0.
16–31	PT	Pause time value. Represents the 16-bit pause quanta (that is, 512 bit times). This pause value is used as part of the pause frame to be sent when TCTRL[TFC_PAUSE] is set. See Section 21.6.2.6, “Flow Control,” for more information.

21.5.3.1.6 DMA Control Register (DMACTRL)

DMACTRL is writable by the user to configure the DMA block. Figure 21-10 describes the definition for the DMACTRL register.

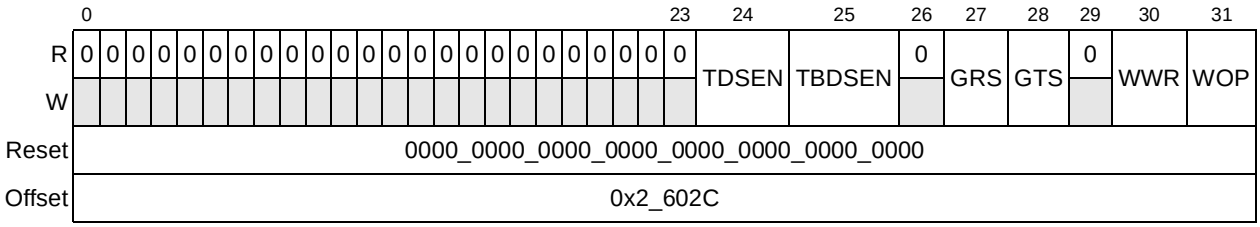


Figure 21-10. DMACTRL Register Definition

Table 21-9 describes the fields of the DMACTRL register.

Table 21-9. DMACTRL Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TDSEN	Tx data snoop enable 0 Disables snooping of all transmit frames from memory. 1 Enables snooping of all transmit frames from memory.
25	TBDSEN	TxBD snoop enable 0 Disables snooping of all transmit BD memory accesses. 1 Enables snooping of all transmit BD memory accesses.
26	—	Reserved

Table 21-9. DMACTRL Field Descriptions (continued)

Bits	Name	Description
27	GRS	Graceful receive stop. If this bit is set, the Ethernet controller stops receiving frames following completion of the frame currently being received. (That is, after a valid end of frame was received). The buffer of the receive frame associated with the EOF is closed and the IEVENT[GRSC] is set. Because the MAC's receive enable bit (MACCFG[Rx_EN]) may still be set, the MAC may continue to receive but the FEC ignores the receive data until GRS is cleared. If this bit is cleared, the FEC scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter) and the first valid frame received uses the next RxBD. If GRS is set, the user must monitor the graceful receive stop complete (GRSC) bit in the IEVENT register to insure that the graceful receive stop was completed. The user can then clear IEVENT[GRSC] and can write to receive registers that are accessible to both user and the FEC hardware without fear of conflict. 0 The FEC scans the input data stream for valid frames. 1 The FEC stops receiving frames following completion of current frame.
28	GTS	Graceful transmit stop. If this bit is set, the Ethernet controller stops transmission after the completion of any frame that is currently being transmitted and the GTSC interrupt in the IEVENT register is asserted. If frame transmission is not currently underway, the GTSC interrupt is asserted immediately. Once transmission has completed, a 'restart' can be accomplished by clearing GTS. 0 The FEC continues. 1 The FEC stops transmission after completion of current frame.
29	—	Reserved
30	WWR	Write with response. This bit gives the user the assurance that a BD was updated in memory before it receives an interrupt concerning a transmit or receive frame. 0 Do not wait for acknowledgement from system for BD writes before setting IEVENT bits. 1 Before setting IEVENT bits TXB, TXF, TXE, IE, XFUN, LC, CRL/XDA, RXB, RXF, the FEC waits for system acknowledgement that the transmit or receive BD being updated was stored in memory.
31	WOP	Wait or poll. This bit, which is applicable only to the transmitter, provides the user the option for the FEC to periodically poll TxBD or to wait for software to tell FEC to fetch a buffer descriptor. While operating in the wait mode, the FEC allows two additional reads of a descriptor which is not ready before entering a halt state. No interrupt is driven (IEVENT[TXE] is clear). To resume transmission, software must clear TSTAT[THLT]. Note that if this bit is set, the user must ensure that all TxBDs involved with sending a frame have their ready bits set before any transmission begins. Otherwise, FEC behavior is boundedly undefined. In wait mode (DMACTRL[WOP] is set) when FEC is processing a frame and an intermediate TxBD's ready bit is cleared, FEC does not halt, but instead continuously polls the same TxBD until the TxBD becomes ready or an Ethernet interface error or a memory error is encountered. If the TxBD becomes ready (TxBD[Ready] is set), FEC continues to process the frame. If an error occurs before the TxBD becomes ready, FEC reports the error in both the IEVENT register as well as the TxBD for Ethernet interface errors, or the IEVENT[EBERR] for a memory error and sets TSTAT[THLT]. **Note**: Software must eventually set all of its TxBDs for a frame, because FEC continuously reads an intermediate TxBD until it becomes ready if insufficient data has been read to surpass the FIFO transmit threshold (FIFO_TX_THR) register value. 0 Poll TxBD every 512 clocks. 1 Do not poll, but wait for a write to TSTAT[THLT].

21.5.3.2 FEC FIFO Control and Status Registers

The following registers allow the user to change some of the default settings in the FIFO that can be used to optimize operation for performance or for safety. They must be set carefully in order to

avoid an underrun condition. Underrun is an error condition in which data is not retrieved from external memory quickly enough, leaving the TX FIFO empty before the complete frame is transmitted. Because different combinations of events, several of which are determined by the user, can lead to underrun, the FEC provides several FIFO registers that allow the user to select the proper setting to be able to tune the system and obtain the maximum performance with minimal chance of underrun. The principal causes for underrun in the FEC are:

- Misaligned data buffer addresses
- Small data buffer sizes
- Combinations of the above

It is recommended that the minimum size data buffers be 64 bytes and that data buffers be 64-byte aligned. The user can deviate from these recommended values to try to increase performance or to use less memory, but unless the default values of some of the FIFO registers are adjusted, the probability of an underrun may also increase. The FIFO_TX_THR (default is 512 entries) 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) 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 512 entries) contains the watermark level to be used for exiting the starve state. These registers are intended to allow the user to make the proper trade-off. If triggered, the starve mode, for instance, automatically raises the priority of FEC fetches from memory.

21.5.3.2.1 FIFO Pause Control Register (FIFO_PAUSE_CTRL)

FIFO_PAUSE_CTRL, shown in Figure 21-11, is writable by the user to configure the properties of the FEC FIFO.

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

	16													29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TFC_PAUSE_EN	0
W																
Reset	0000_0000_0000_0000															
Offset	0x2_604C															

Figure 21-11. FIFO_PAUSE_CTRL Register Definition

Table 21-10 describes the fields of the FIFO_PAUSE_CTRL register.

Table 21-10. FIFO_PAUSE_CTRL Field Descriptions

Bits	Name	Description
0–29	—	Reserved.
30	TFC_PAUSE_EN	TFC_PAUSE enable. This bit enables the ability to transmit a pause control frame by setting the TCTRL[TFC_PAUSE] bit. This bit is cleared at reset. 0 Pause control frame transmission disabled. 1 Pause control frame transmission enabled.
31	—	Reserved.

21.5.3.2.2 FIFO Transmit Threshold Register (FIFO_TX_THR)

The purpose of the threshold register is to trigger the unloading of FIFO data to the PHY. It represents the numerical SRAM entry (0-255 for a 1-Kbyte FIFO) to trigger the threshold function. If the used entry count is equal to or greater than the threshold register, then transmission can begin. This register is read/write by software. This register is initialized to 0000_0000_0000_0000_0000_0000_1000_0000 at system reset. Figure 21-12 describes the definition for the FIFO_TX_THR register.

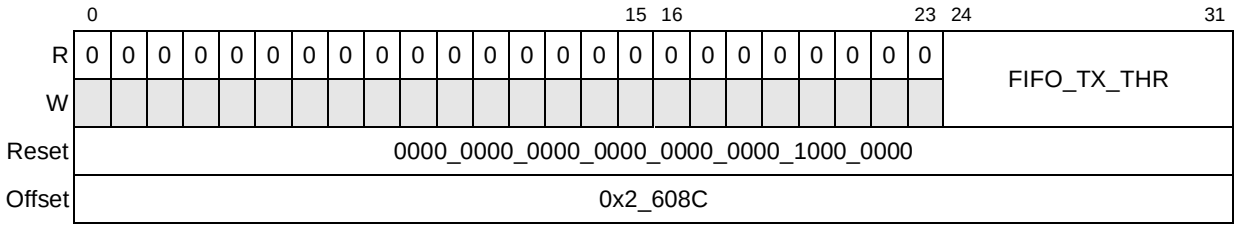


Figure 21-12. FIFO_TX_THR Register Definition

Table 21-11 describes the fields of the FIFO_TX_THR register.

Table 21-11. FIFO_TX_THR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	FIFO_TX_THR	FIFO transmit threshold. Marks the number of entries in the transmit FIFO that when reached triggers the unloading of frame data into the MAC. If the FIFO Tx used entry count is equal to or greater than FIFO_TX_THR, frame transmission request is sent to the MAC.

21.5.3.2.3 FIFO Transmit Starve Register (FIFO_TX_STARVE)

The purpose of the starve register is to inform the system of extremely imminent underrun conditions. It represents the numerical SRAM entry (0–255 for 1-Kbyte FIFO) to trigger the starve function. If the used entry count in the FIFO is less than or equal to the starve register, a starve alert is triggered. This triggered condition is used to throttle back the OCN2CU interface at the e500 coherency module (ECM), to change the Tx transaction priority. This register is read/write by software. Figure 21-13 describes the definition for the FIFO_TX_STARVE register.

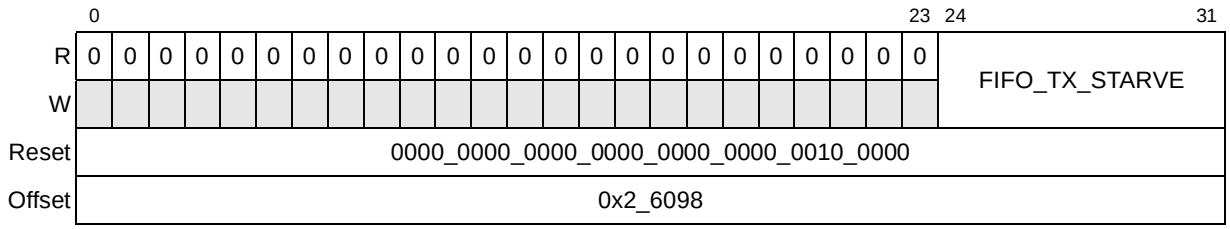


Figure 21-13. FIFO_TX_STARVE Register Definition

Table 21-12 describes the fields of the FIFO_TX_STARVE register.

Table 21-12. FIFO_TX_STARVE Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	FIFO_TX_STARVE	FIFO transmit starve. Indicates the value to trigger the transmit starve function. It triggers once the FIFO transmit used entry count is less than or equal to the FIFO Tx starve. The starve state turns off when the FIFO Tx used entry count becomes greater than or equal to the FIFO Tx starve shutoff register.

21.5.3.2.4 FIFO Transmit Starve Shutoff Register (FIFO_TX_STARVE_SHUTOFF)

The starve shutoff registers contains the watermark level to be used for coming out of the starve state. If the starve state is in effect and the number of valid entries in the FIFO becomes greater than or equal to the value in the FIFO_TX_STARVE_SHUTOFF register, the starve condition

ends. This register is read/write by software. Figure 21-14 describes the definition for the FIFO_TX_STARVE_SHUTOFF register.

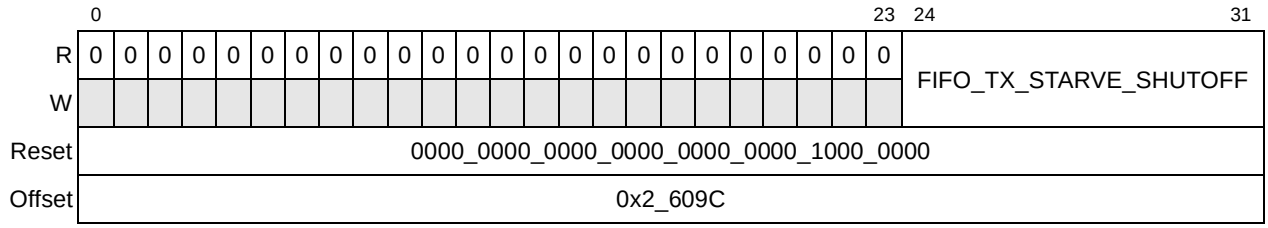


Figure 21-14. FIFO_TX_STARVE_SHUTOFF Register Definition

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

Table 21-13. FIFO_TX_STARVE_SHUTOFF Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–31	FIFO_TX_STARVE_SHUTOFF	FIFO transmit starve shutoff. Indicates the value to use to turn off the starve state. The starve state turns off when the FIFO Rx used entry count becomes greater than or equal to the FIFO Rx starve shutoff register.

21.5.3.3 FEC Transmit Control and Status Registers

This section describes the control and status registers that are used specifically for transmitting Ethernet frames. All of the registers are 32 bits wide.

21.5.3.3.1 Transmit Control Register (TCTRL)

This register is writable by the user to configure the transmit block. Figure 21-15 describes the definition for the TCTRL register.

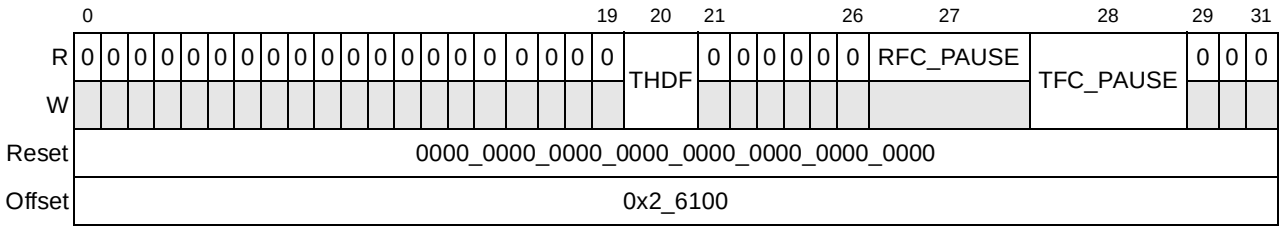


Figure 21-15. TCTRL Register Definition

Table 21-14 describes the fields of the TCTRL register.

Table 21-14. TCTRL Field Descriptions

Bits	Name	Description
0–19	—	Reserved
20	THDF	Transmit half-duplex flow control. Written by user. This bit is not self-resetting. 0 Disable back pressure. 1 Back pressure is applied to media.
21–26	—	Reserved
27	RFC_PAUSE	Receive flow control pause frame. Written by the FEC. This read-only status bit is set if a flow control pause frame was received and the transmitter is paused for the duration defined in the received pause frame. This bit automatically clears after the pause duration is complete. 0 Pause duration complete. 1 Pause duration in progress.
28	TFC_PAUSE	Transmit flow control pause frame. Use this bit to transmit a PAUSE frame. To transmit a flow control pause frame, first set FIFO_PAUSE_CTRL[TFC_PAUSE_EN]. Next, set MACCFG1[GTS]. 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

21.5.3.3.2 Transmit Status Register (TSTAT)

This register is written by the FEC to convey DMA status information. Figure 21-16 describes the definition for the TSTAT register.

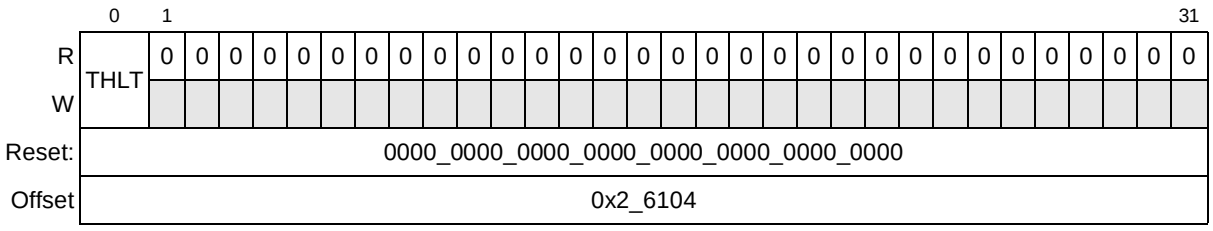


Figure 21-16. TSTAT Register Definition

Table 21-15 describes the fields of the TSTAT register.

Table 21-15. TSTAT Field Descriptions

Bit	Name	Description
0	THLT	Transmit halt 0 No hardware initiated transmission halt. 1 FEC transmission function halted. Set by the FEC to inform the user that it is no longer processing transmit frames and the transmit DMA function is disabled by hardware. To re-start the transmission function, the user must clear this bit by writing a one.
1–31	—	Reserved

21.5.3.3.3 TxBD Data Length Register (TBDLEN)

TBDLEN is a DMA register that contains the number of bytes remaining in the current transmit buffer. Figure 21-17 describes the definition for the TBDLEN register.

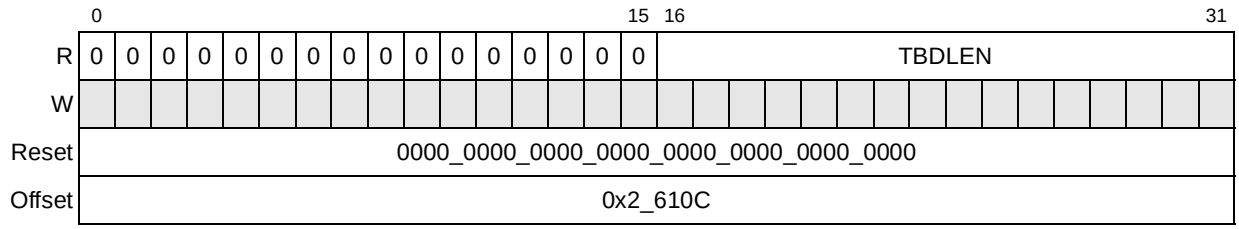


Figure 21-17. TBDLEN Register Definition

Table 21-16 describes the fields of the TBDLEN register.

Table 21-16. TBDLEN Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	TBDLEN	Internally written by the DMA module. The transmit channel remains active until TBDLEN is 0.

21.5.3.3.4 Current Transmit Buffer Descriptor Pointer Register (CTBPTR)

CTBPTR, shown in Figure 21-18, contains the address of the transmit buffer descriptor either currently being processed, or processed most recently.

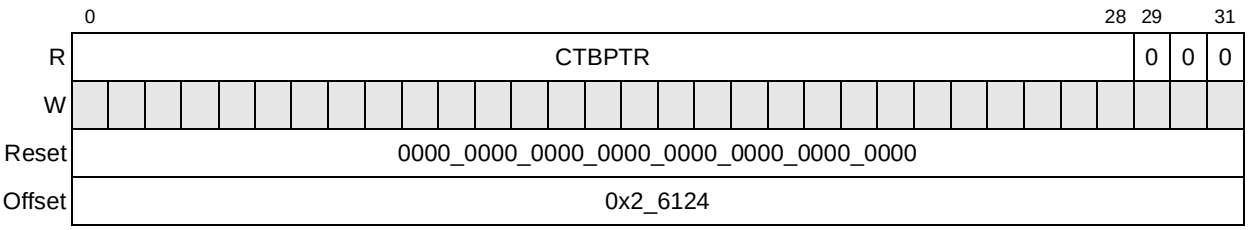


Figure 21-18. CTBPTR Register Definition

Table 21-17 describes the fields of the CTBPTR register.

Table 21-17. CTBPTR Field Descriptions

Bits	Name	Description
0–28	CTBPTR	Internally written by the DMA module. The value increments by eight (bytes) each time a descriptor is read from memory.
29–31	—	Reserved

21.5.3.3.5 Transmit Buffer Descriptor Pointer Register (TBPTR)

TBPTR contains the low-order 32 bits of the next transmit buffer descriptor address. Figure 21-19 describes the definition for the TBPTR register. This register takes on the value of TBASE when the TBASE register is written by software.

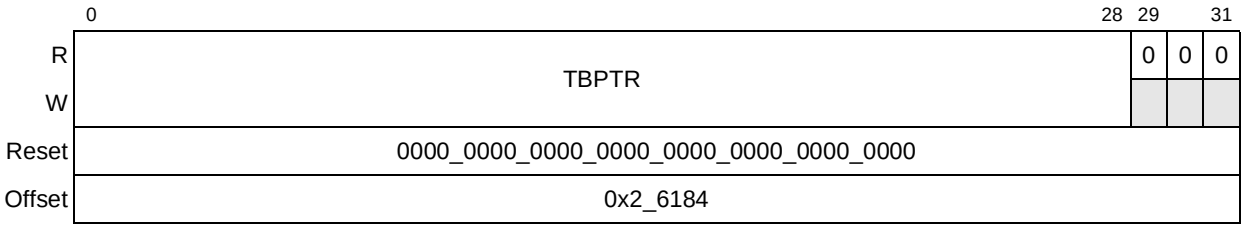


Figure 21-19. TBPTR Register Definition

Table 21-18 describes the fields of the TBPTR register.

Table 21-18. TBPTR Field Descriptions

Bit	Name	Description
0–28	TBPTR	Internally written by the DMA module. The value increments by eight (bytes) each time a descriptor is read from memory.
29–31	—	Reserved

21.5.3.3.6 Transmit Descriptor Base Address Register (TBASE)

The TBASE register is written by the user with the TxDB base address. The value must be divisible by eight for the 8-byte data buffer descriptors. Figure 21-20 describes the definition for the TBASE register.

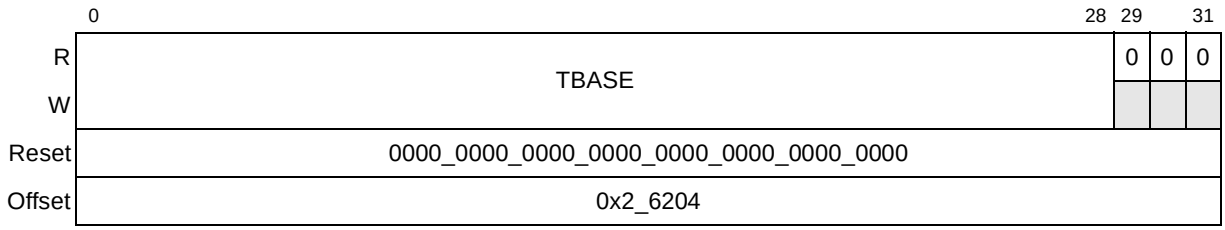


Figure 21-20. TBASE Register Definition

Table 21-19 describes the fields of the TBASE register.

Table 21-19. TBASE Field Descriptions

Bits	Name	Description
0–28	TBASE	Transmit base. Defines the starting location in the memory map for the FEC TxBDs. This field must be 8-byte aligned. Together with setting the W (wrap) bit in the last BD, the user can select how many BDs to allocate for the transmit packets. The user must initialize TBASE before enabling the FEC transmit function.
29–31	—	Reserved

21.5.3.3.7 Out-of-Sequence TxBD Register (OSTBD)

The out-of-sequence TxBD register (OSTBD), shown in Figure 21-21, includes the status/control and data length in the same format as a regular TxBD. It is useful for sending flow control frames. The OSTBD[R] is always checked between frames. If it is not ready, a regular frame is sent. 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.

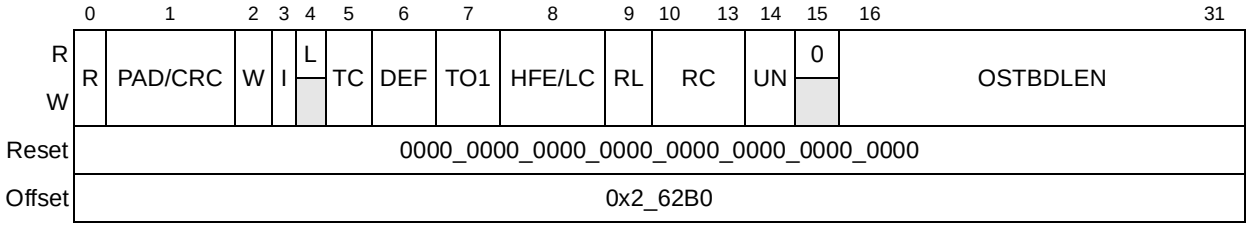


Figure 21-21. OSTBD Register Definition

Table 21-20 describes the fields of the OSTBD register.

Table 21-20. OSTBD Field Descriptions

Bits	Name	Description
0	R	Ready. Written by both the FEC and the user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer was transmitted or after an error condition is encountered. 1 The data buffer, 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] 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, FEC always pads up to the IEEE minimum frame length of 64 bytes.

Table 21-20. OSTBD Field Descriptions (continued)

Bits	Name	Description
2	W	Wrap. Written by user. This bit is ignored by FEC.
3	I	Interrupt. Written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[TXF] is set after this buffer is serviced. Setting 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 TxBD[PAD/CRC] is set. 1 Transmit the CRC sequence after the last data byte.
6	DEF	Defer indication. Written by the FEC. Hardware updates this bit after transmitting a frame when used as a ‘defer indicator.’ Software/user updates this bit while building a transmit buffer descriptor 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 the FEC. 0 Transmission before maximum retry limit is hit. 1 The transmitter failed (maximum retry limit + 1 attempts to successfully send a message due to repeated collisions). The FEC terminates the transmission and updates RL.
10–13	RC	Retry count. Written by the FEC. 0 The frame is sent correctly the first time. 1 More than zero attempts where needed to send the transmit frame. When this field is 15, 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
14	UN	Underrun. Written by the FEC. 0 No underrun encountered (data was retrieved from external memory in time to send a complete frame). 1 The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer. The FEC terminates the transmission and updates UN.
15	—	Reserved
16–31	OSTBDLEN	Out-of-sequence TxBD data length. Written by user. Data length is the number of octets the FEC transmits from this BD’s data buffer. It is never modified by the FEC.

21.5.3.3.8 Out-of-Sequence Tx Data Buffer Pointer Register (OSTBDP)

The out-of-sequence Tx data buffer pointer register (OSTBDP), shown in Figure 21-22, is written by the user and contains the data buffer pointer fields in the same format as a regular TxBD and, together with OSTBD, provides the complete 8-byte descriptor. This area must be cleared while not in use.

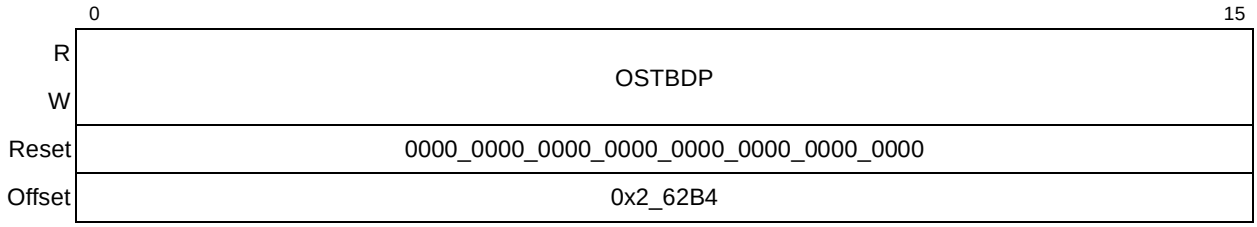


Figure 21-22. OSTBDP Register Definition

Table 21-21 describes the field of the OSTBDP register.

Table 21-21. OSTBDP Field Descriptions

Bita	Name	Description
0–31	OSTDBP	Out-of-sequence Tx data buffer pointer. Written by user. The Tx data buffer pointer contains the address of the associated data buffer. There are no alignment requirements for this address.

21.5.3.4 FEC Receive Control and Status Registers

This section describes the control and status registers that are used specifically for receiving Ethernet frames. All of the registers are 32 bits wide.

21.5.3.4.1 Receive Control Register (RCTRL)

The RCTRL, shown in Figure 21-23, controls the operational mode of the receive block. It must be written only after a system reset (at initialization) or if DMACTRL[GRS] is zero.

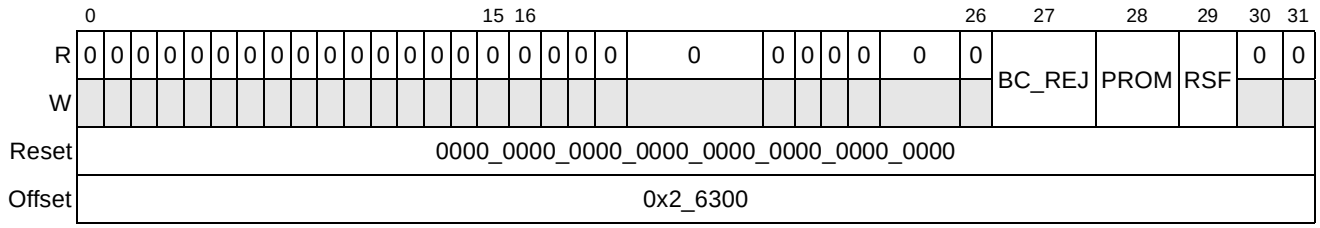


Figure 21-23. RCTRL Register Definition

Table 21-22 describes the fields of the RCTRL register.

Table 21-22. RCTRL Field Descriptions

Bits	Name	Description
0–26	—	Reserved
27	BC_REJ	Broadcast frame reject. If this bit is set, frames with DA (destination address) = 0xFFFF_FFFF_FFFF are rejected unless RCTRL[PROM] is set. If both BC_REJ and PROM are set, then frames with broadcast DA are accepted and the M (MISS) bit is set in the receive BD. 0 Broadcast frame reject is disabled. 1 Broadcast frame reject is enabled.
28	PROM	Promiscuous mode. When set, all frames except pause frames are accepted. 0 Promiscuous mode is disabled. 1 Promiscuous mode is enabled.
29	RSF	Receive short frame mode. When set, enables the reception of frames shorter than MINFLR bytes. Note that in order for short frames to be received when RSF is set, a DA hit must occur. When RSF is cleared, all frames shorter than MINFLR are automatically rejected. 0 Receive short frame mode is disabled. 1 Receive short frame mode is enabled.
30–31	—	Reserved

21.5.3.4.2 Receive Status Register (RSTAT)

The FEC writes to this register under the following conditions:

- The receiver runs out of descriptors.
- The receiver was halted because an error condition was encountered while receiving a frame.

Software must clear the QHLT bit to take the FEC's receiver function out of a halt state. Figure 21-24 describes the definition for the RSTAT register.

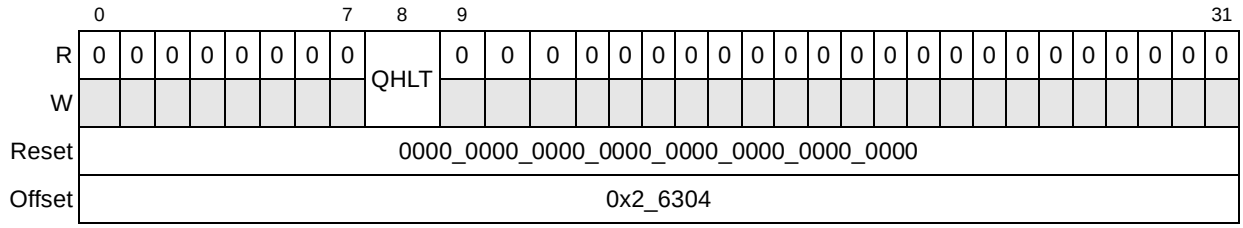


Figure 21-24. RSTAT Register Definition

Table 21-23 describes the fields of the RSTAT register.

Table 21-23. RSTAT Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8	QHLT	RxBD queue is halted. When IEVENT[BSY] or IEVENT[EBERR] is set during reception of a packet, RSTAT[QHLT] is also set. In order to begin receiving packets again, the user must clear RSTAT[QHLT]. This bit is set whenever the FEC reads an RxBD with its Empty field cleared or encounters a system bus error while processing an RX packet. It is a hardware-initiated stop indication (DMA_CTRL[GRS] being set by the user does not cause this bit to be set). The current frame and all other frames directed to the halted queue are discarded. A write with a value of 1 re-enables the queue for receiving. 0 RxBD queue is enabled for Ethernet reception. (That is, it is not halted.) 1 All Ethernet controller receive activity to RxBD queue is halted.
9–31	—	Reserved

21.5.3.4.3 RxBD Data Length Register (RBDLEN)

RBDLEN is a DMA register that contains the number of bytes remaining in the current receiver buffer. Figure 21-25 describes the definition for the RBDLEN register.

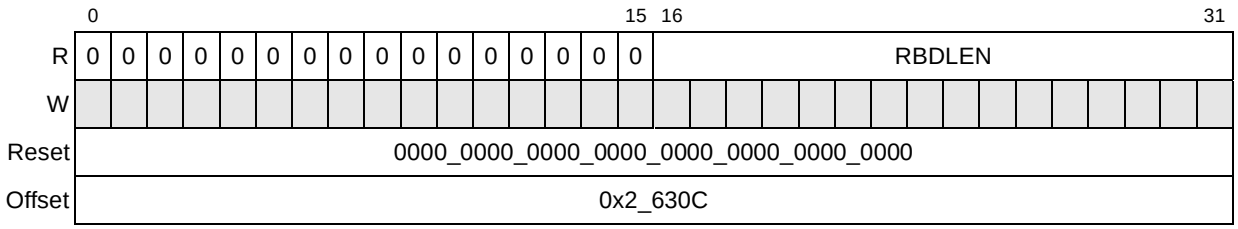


Figure 21-25. RBDLEN Register Definition

Table 21-24 describes the fields of the RBDLEN register.

Table 21-24. RBDLEN Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	RBDLEN	The RBDLEN is internally written by the DMA module. If RBDLEN is cleared, all activity in the receive channel stops.

21.5.3.4.4 Current Receive Buffer Descriptor Pointer Register (CRBPTR)

CRBPTR, shown in Figure 21-26, contains the address of the receive buffer descriptor either currently being processed, or processed most recently.

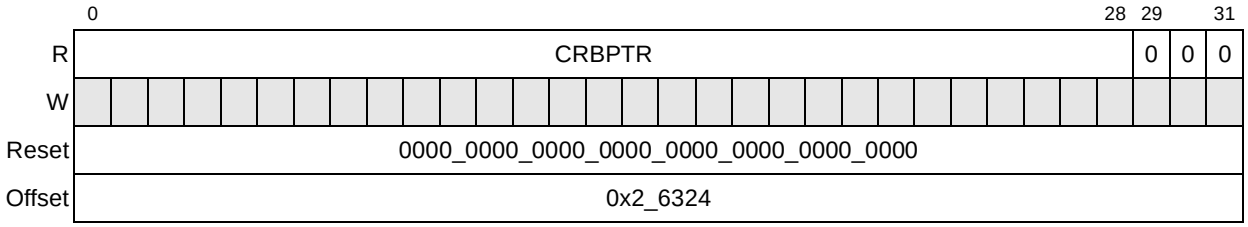


Figure 21-26. CRBPTR Register Definition

Table 21-25 describes the fields of the CRBPTR register.

Table 21-25. CRBPTR Field Descriptions

Bits	Name	Description
0–28	CRBPTR	The CRBPTR register is internally written by the DMA module. The value increments by eight (bytes) each time a descriptor is read from memory.
29–31	—	Reserved

21.5.3.4.5 Maximum Receive Buffer Length Register (MRBLR)

The MRBLR, shown in Figure 21-27, is written by the user. It informs the FEC how much space is in the receive buffer pointed to by the RxBD.

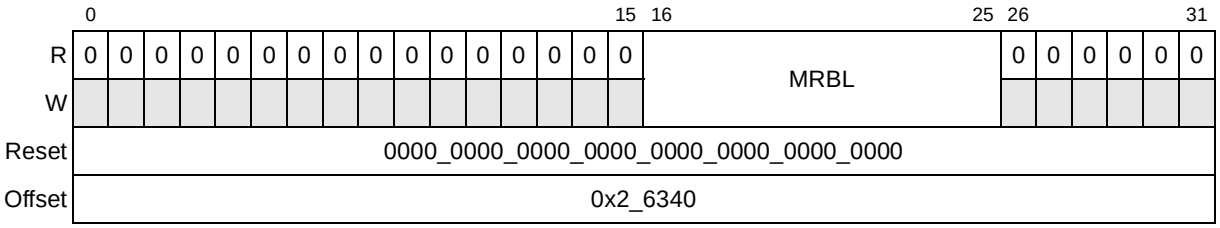


Figure 21-27. MRBLR Register Definition

Table 21-26 describes the fields of the MRBLR register.

Table 21-26. MRBLR Field Descriptions

Bits	Name	Description
0–15	—	To ensure that MRBL is a multiple of 64 these bits are reserved and must be cleared.
16–25	MRBL	Maximum receive buffer length. The number of bytes that the FEC receiver writes to the receive buffer. The MRBL register is written by the user with a multiple of 64 for all modes. The FEC can write fewer bytes to the buffer than the value set in MRBL if a condition such as an error or end-of-frame condition occurs, but it never exceeds the MRBL value; therefore, user-supplied buffers must be at least as large as the MRBL. Note that transmit buffers can have varying lengths by programming TxBD[Data Length], as needed, and are unaffected by the MRBLR value. MRBLR is not intended to be changed dynamically while FEC is operating. Change MRBLR only when the FEC receive function is disabled.
26–31	—	To ensure that MRBL is a multiple of 64 these bits are reserved and must be cleared.

21.5.3.4.6 Receive Buffer Descriptor Pointer Register (RBPTR)

RBPTR, shown in Figure 21-28, contains the receive buffer descriptor address. It takes on the value of RBASE when the RBASE register is written by software.

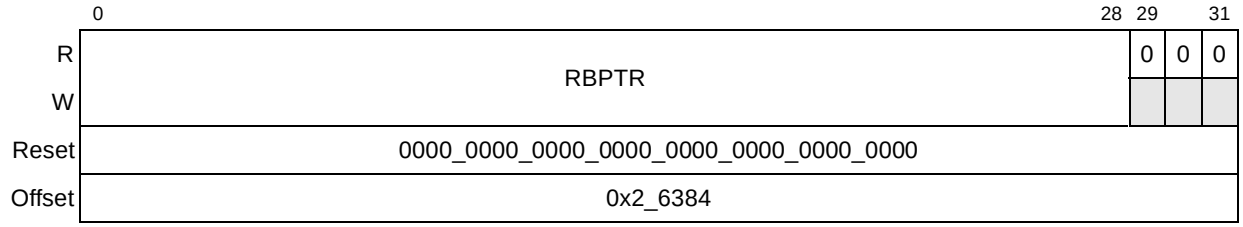


Figure 21-28. Receive Buffer Descriptor Pointer Definition

Table 21-27 describes the fields of the RBPTR register.

Table 21-27. RBPTR Field Descriptions

Bits	Name	Description
0–28	RBPTR	The RBPTR register is internally written by the DMA module. The value increments by eight (bytes) each time a descriptor is read from memory.
29–31	—	Reserved

21.5.3.4.7 Receive Descriptor Base Address Register (RBASE)

RBASE, shown in Figure 21-29, is written by the user with the RxBD base address and must be divisible eight for the 8-byte buffer descriptors.

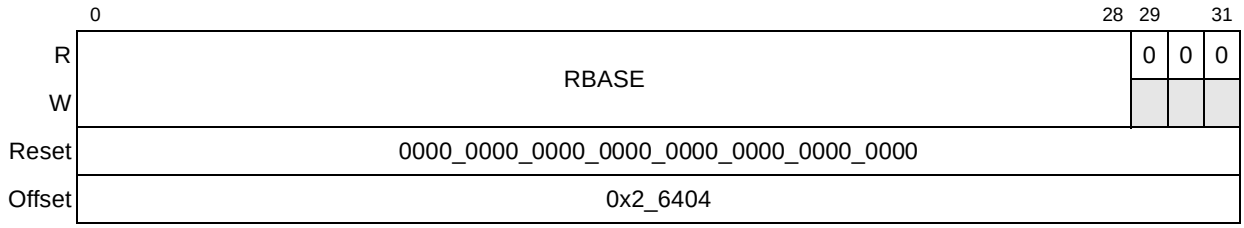


Figure 21-29. RBASE Register Definition

Table 21-28 describes the fields of the RBASE register.

Table 21-28. RBASE Field Descriptions

Bits	Name	Description
0–28	RBASE	Receive base. RBASE defines the starting location in the memory map for the FEC RxBD. This field must be 8-byte aligned.
29–31	—	Reserved

21.5.3.5 MAC Functionality

This section describes the MAC registers and provides a brief overview of some of the functionality that can be exercised through the use of these registers, particularly those that provide functionality not explicitly required by the IEEE 802.3 standard. All of the MAC registers are 32 bits wide.

21.5.3.5.1 Configuring the MAC

MAC configuration registers 1 and 2 provide a way to configure the MAC as follows:

- 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 – Two different 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]). These options include appending a valid CRC, or padding and then appending a valid CRC, resulting in a minimum frame of 64 octets. In addition to the programmable register set, the pad/CRC behavior can be dynamically adjusted on a per-packet basis.

21.5.3.5.2 Controlling CSMA/CD

The half-duplex register (HAFDUP) controls the carrier-sense multiple access/collision detection (CSMA/CD) logic. After a packet is sent, the device begins timing the inter-packet gap (IPG) as programmed in the back-to-back IPG configuration register. The system is then free to begin another frame transfer.

In full-duplex mode, the carrier sense (FEC_CRS) and collision (FEC_COL) indications from the PHY are ignored, but in half-duplex mode, the FEC defers to FEC_CRS and, following a carrier event, times the IPG using the non–back-to-back IPG configuration values that include support for the optional two-thirds, one-third FEC_CRS deferral process. This optional IPG mechanism enhances system robustness and ensures fair access to the medium. During the first two-thirds of the IPG, the IPG timer is cleared if FEC_CRS is sensed. During the final one-third of the IPG, FEC_CRS is ignored and the transmission begins when IPG is timed. The two-thirds/one-third ratio is recommended.

21.5.3.5.3 Handling Packet Collisions

When transmitting a packet in half-duplex mode, the FEC is sensitive to FEC_COL. If a collision occurs, it aborts the packet and outputs the 32-bit jam sequence. The jam sequence is made up of several bits of the CRC, inverted to guarantee an invalid CRC upon reception. A signal is sent to the system indicating a collision occurred and the start of the frame is needed for retransmission. The FEC then backs-off of the medium for a time determined by the truncated binary exponential

back-off (BEB) algorithm. Following this back-off time, the packet is retried. The back-off time can be skipped if configured 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 flushes the frame and move to the next one in line. If the system requests to send a packet while the FEC is deferring to a carrier, the FEC simply waits until the end of the carrier event and the timing of IPG before it honors the request.

When packet transmission attempts experience collisions, the FEC outputs the jam sequence and waits some amount of time before retrying the packet. This amount of time is determined by a controlled randomization process called truncated binary exponential back-off. The amount of time is an integer number of slot times. The number of slot times to delay before the *n*th retransmission attempt is chosen as a uniformly distributed random integer *r* in the range:

$$0 \leq r \leq 2^k, \text{ where } k = \min(n,10).$$

So, after the first collision, FEC backs-off either 0 or 1 slot times. After the fifth collision, the FEC backs-off between 0 and 32 slot times. After the tenth collision, the maximum number of slot times to back-off is 1024. This can be adjusted via the half-duplex register. An alternate truncation point, such as 7 for instance, can be programmed. On average, the MAC is be more aggressive after 7 collisions than other stations on the network.

21.5.3.5.4 Controlling Packet Flow

Packet flow can be dealt with in a number of ways with the FEC. A default retransmit attempt limit of 15 can be reduced using the half-duplex register. The slot time or collision window can be used to gate the retry window and possibly reduce the amount of transmit buffering within the system. The slot time for 10/100 Mbps is 512 bit times. Since the slot time begins at the beginning of the packet, the end occurs around the 56th byte of the frame data.

Full-duplex flow control is provided for in IEEE 802.3x. Currently the standard does not address flow control in half-duplex environments. Common in the industry, however, is the concept of back pressure. The FEC implements the optional back pressure mechanism using the raise carrier method. When the system receive logic wants to stop the reception of packets in a network-friendly way, transmit half-duplex flow control (THDF) is asserted (TCTRL[THDF]). If the medium is idle, the FEC raises carrier by transmitting preamble. Other stations on the half-duplex network then defer to the carrier.

In the event the preamble transmission happens to cause a collision, the FEC ensures the minimum 96-bit presence on the wire, then drops preamble and waits a back-off time depending on the value of the configuration bit back pressure no back-off (half-duplex) [BPNB]. These transmitting-preamble-for-back pressure collisions are not counted. When BPBN is asserted, the FEC waits an inter-packet gap before resuming the transmission of preamble following the collision and does not defer. If deasserted, the FEC adheres to the truncated BEB algorithm that allows packets to be received. This also can be detrimental in that packets can now experience excessive collisions causing them to be dropped in the stations from which they originate. To

reduce the likelihood of lost packets and packets leaking through the back pressure mechanism, BPNB must be set.

The FEC drops carrier (ceases transmitting preamble) periodically to avoid excessive defer conditions in other stations on the shared network. If while applying back pressure, the FEC is requested to send a packet, it stops sending preamble, and waits one IPG before sending the packet. BPNB applies for any collision that occurs during the sending of this packet. Collisions for packets while THBP is asserted are counted. The FEC does not defer while attempting to send packets while in back pressure. Again, back pressure is nonstandard, but can reduce the flow of receive packets.

21.5.3.6 MAC Registers

This section contains descriptions of the MAC registers.

21.5.3.6.1 MAC Configuration Register 1 (MACCFG1)

The MACCFG1, shown in Figure 21-30, is written by the user.

	0	1	2								11	12	13	14	15
R	Soft_Reset	0	0	0	0	0	0	0	0	0	0	Reset_ Rx_MC	Reset_ Tx_MC	Reset_ Rx_Fun	Reset_ Tx_Fun
W															
Reset	0000_0000_0000_0000														

	16						22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	Loop_ Back	0	0	Rx_Flow	Tx_Flow	Sync'd_ Rx_EN	Rx_EN	Sync'd_ Tx_EN	Tx_EN
W																
Reset	0000_0000_0000_0000															
Offset	0x2_6500															

Figure 21-30. MACCFG1 Register Definition

Table 21-29 describes the fields of the MACCFG1 register.

Table 21-29. MACCFG1 Field Descriptions

Bits	Name	Description
0	Soft_Reset	0 Default 1 Setting this bit puts all modules within the MAC in reset.
1–11	—	Reserved
12	Reset_Rx_MC	0 Default 1 Setting this bit puts the receive MAC control block in reset. This block detects control frames and contains the pause timers.

Table 21-29. MACCFG1 Field Descriptions (continued)

Bits	Name	Description
13	Reset_Tx_MC	0 Default 1 Setting this bit puts the PETMC transmit MAC control block in reset. This block multiplexes data and control frame transfers. It also responds to XOFF PAUSE control frames.
14	Reset_Rx_Fun	0 Default 1 Setting this bit puts the receive function block in reset. This block performs the receive frame protocol.
15	Reset_Tx_Fun	0 Default 1 Setting this bit puts the transmit function block in reset. This block performs the frame transmission protocol.
16–22	—	Reserved
23	Loop_Back	0 Default. 1 Causes the MAC transmit outputs to be looped back to the MAC receive inputs.
24–25	—	Reserved
26	Rx_Flow	0 Default. The receive MAC control ignores PAUSE flow control frames. 1 The receive MAC control can detect and act on PAUSE flow control frames.
27	Tx_Flow	0 Default. The transmit MAC control cannot send flow control frames. 1 The transmit MAC control can send PAUSE flow control frames when requested by the system.
28	Sync'd_Rx_EN	Receive enable synchronized to the receive stream. (Read only) 0 Frame reception 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 Default. The MAC cannot receive frames from the PHY. 1 The MAC can 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 Default. Clearing this bit prevents the transmission of frames. 1 Setting this bit allows the MAC to transmit frames from the system.

21.5.3.6.2 MAC Configuration Register 2 (MACCFG2)

The MACCFG2, shown in Figure 21-31, is written by the user.

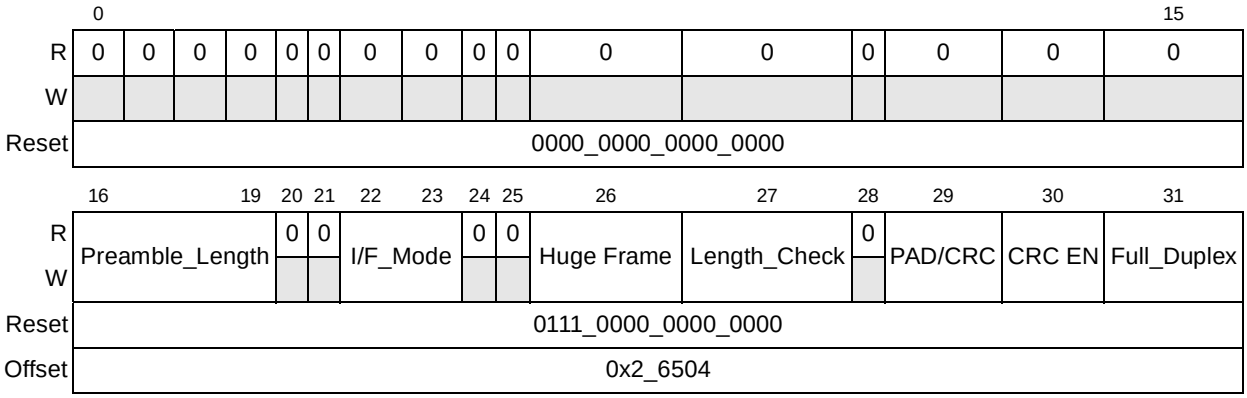


Figure 21-31. MACCFG2 Register Definition

Table 21-30 describes the fields of the MACCFG2 register.

Table 21-30. MACCFG2 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–19	Preamble_Length	This field determines the length in bytes of the preamble field of the packet. Its default is 0x7. A preamble length of 0 is not supported.
20–21	—	Reserved
22–23	I/F_Mode	Determines the type of interface the MAC is connected to. 00 Reserved 01 Nibble mode (MII) 10 Reserved 11 Reserved
24–25	—	Reserved
26	Huge Frame	0 Default. The MAC limits the length of frames at the MAXIMUM FRAME LENGTH value. 1 Frames longer than the MAXIMUM FRAME LENGTH can be transmitted and received.
27	Length_Check	0 Default. Clear this bit if no length field checking is desired. 1 The MAC checks the frame's length field to ensure it matches the actual data field length.
28	—	Reserved
29	PAD/CRC	0 Default. Transmitted short frames are not padded. 1 The MAC pads all transmitted short frames and appends a CRC to every frame whether or not padding was required.
30	CRC EN	CRC enable 0 Default. Clear if frames presented to the MAC have a valid length and contain a valid CRC. 1 Causes the MAC to append a CRC on all frames. If the configuration bit PAD/CRC or the per-packet PAD/CRC is set, CRC EN is ignored.
31	Full_Duplex	0 Default. Clearing this bit configures the MAC to operate in half-duplex mode only. 1 Configures the MAC to operate in full-duplex mode.

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

The IPGIFG, shown in Figure 21-32, is written by the user.

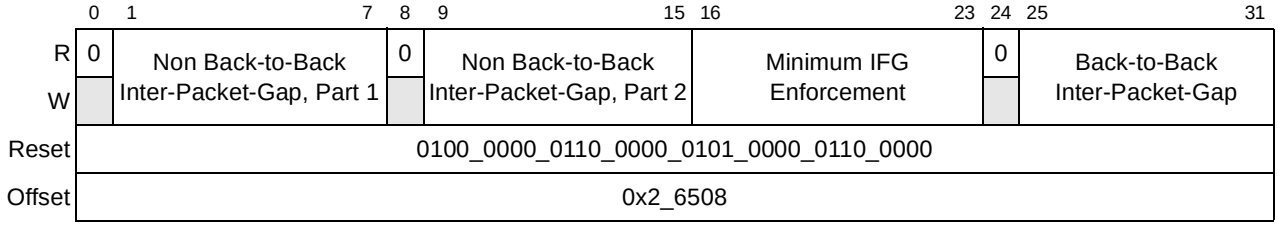


Figure 21-32. IPGIFG Register Definition

Table 21-31 describes the fields of the IPGIFG register.

Table 21-31. IPGIFG Field Descriptions

Bits	Name	Description
0	—	Reserved
1–7	Non Back-to-Back Inter-Packet-Gap, Part 1	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. But if carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision and ensuring fair access to medium. Its range of values is 0x00 to IPGR2. The default, 0x40, follows the two-thirds/one-thirds guideline.
8	—	Reserved
9–15	Non Back-to-Back Inter-Packet-Gap, Part 2	This is a programmable field representing the non-back-to-back inter-packet-gap in bits. The default, 0x60, represents the minimum IPG of 96 bits.
16–23	Minimum IFG Enforcement	Minimum number of bits of IFG to enforce between frames. A frame whose IFG is less than that programmed is dropped. The default setting of 0x50 (80d) represents half of the nominal minimum IFG, which is 160 bits.
24	—	Reserved
25–31	Back-to-Back Inter-Packet-Gap	IPG between back-to-back packets. This IPG parameter is used exclusively in full-duplex mode and in half-duplex mode when two transmit packets are sent back-to-back. Set this field to the number of bits of IPG desired. The default (minimum) of 0x60 (96d) represents the IPG of 96 bits.

21.5.3.6.4 Half-Duplex Register (HAFDUP)

The HAFDUP register, shown in Figure 21-33, is written by the user.

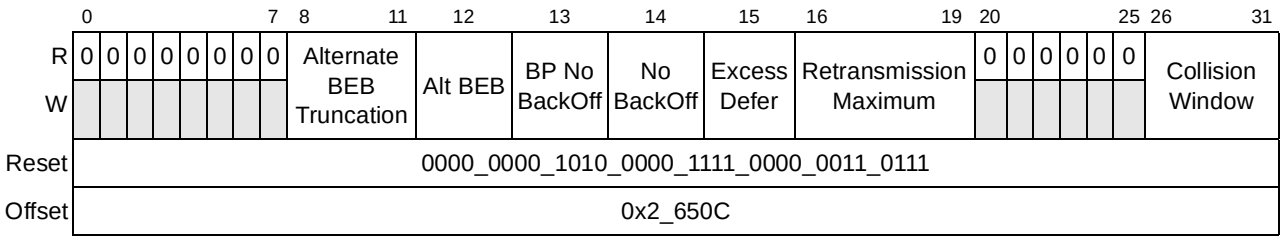


Figure 21-33. Half-Duplex Register Definition

Table 21-32 describes the fields of the HAFDUP register.

Table 21-32. HAFDUP Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	Alternate BEB Truncation	Used when alternate binary exponential back-off enable is set. The value programmed is substituted for the Ethernet standard value of ten.
12	Alt BEB	Setting this bit configures the Tx MAC to use 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 $2^{10} – 1$ as the maximum back-off time. Clearing this bit causes the Tx MAC to follow the standard binary exponential back-off rule.
13	BP No BackOff	Setting this bit configures the Tx MAC to immediately retransmit, following a collision, during back pressure operation. Clearing this bit causes the Tx MAC to follow the binary exponential back-off rule.
14	No BackOff	Setting this bit configures the Tx MAC to immediately retransmit following a collision. Clearing this bit causes the Tx MAC to follow the binary exponential back-off rule.
15	Excess Defer	Setting this bit configures the Tx MAC to allow the transmission of a packet that was excessively deferred. Clearing this bit causes the Tx MAC to abort the transmission of a packet that was excessively deferred.
16–19	Retransmission Maximum	Specifies 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), which is the default.
20–25	—	Reserved
26–31	Collision Window	The slot time or collision window during which collisions occur in properly configured networks. Because the window starts at the start of transmission, the preamble and SFD are included. Its default of 0x37 (55d) corresponds to the count of frame bytes at the end of the window.

21.5.3.6.5 Maximum Frame Length Register (MAXFRM)

The MAXFRM register is written by the user. Figure 21-34 describes the MAXFRM register.

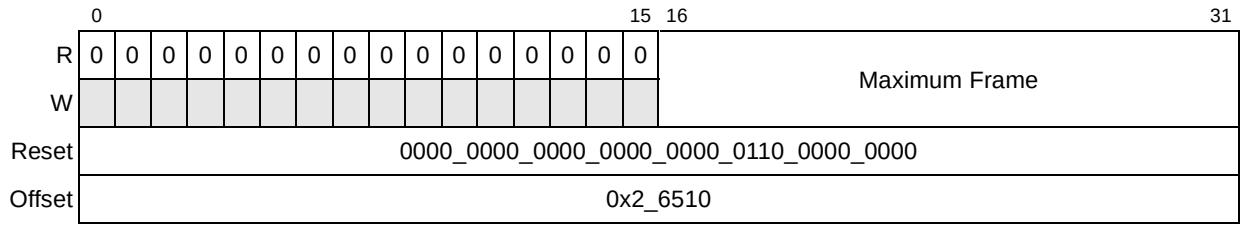


Figure 21-34. Maximum Frame Length Register Definition

Table 21-33 describes the fields of the MAXFRM register.

Table 21-33. MAXFRM Descriptions

Bit	Name	Description
0–15	—	Reserved
16–31	Maximum Frame	Sets the maximum frame size in both the transmit and receive directions. (Refer to MACCFG2[Huge Frame].)

21.5.3.6.6 Interface Status Register (IFSTAT)

Figure 21-35 shows the IFSTAT register.

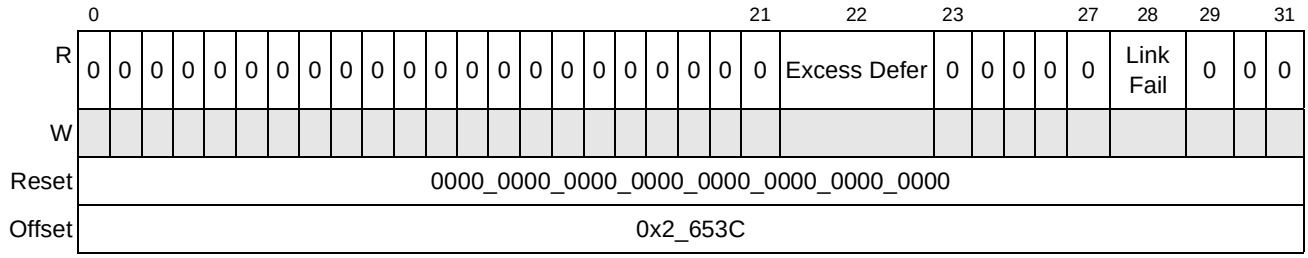


Figure 21-35. Interface Status Register Definition

Table 21-34 describes the fields of the IFSTAT register.

Table 21-34. IFSTAT Field Descriptions

Bit	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

21.5.3.6.7 Station Address Register Part 1 (MACSTNADDR1)

The MACSTNADDR1 register is written by the user. Figure 21-36 describes 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 that the I/G and U/L bits of the frame's DA field are 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.

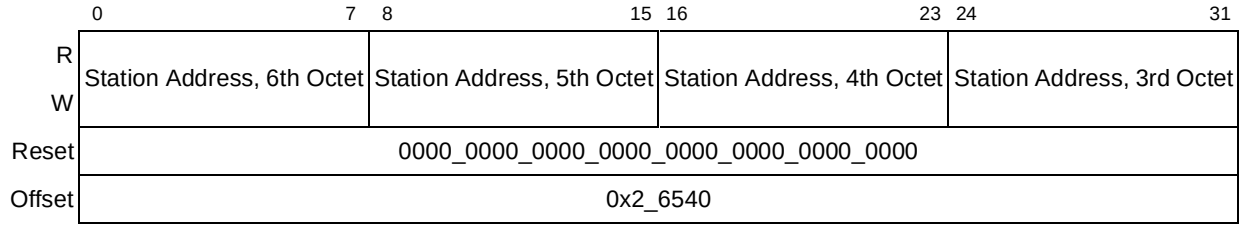


Figure 21-36. Station Address Part 1 Register Definition

Table 21-35 describes the fields of the MACSTNADDR1 register.

Table 21-35. MACSTNADDR1 Field Descriptions

Bit	Name	Description
0–7	Station Address, 6th Octet	This field holds the sixth octet of the station address. The sixth octet (station address bits 40–47) defaults to a value of 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.

21.5.3.6.8 Station Address Register Part 2 (MACSTNADDR2)

The MACSTNADDR2 register is written by the user. Figure 21-37 describes the MACSTNADDR2 register. Note that the I/G and U/L bits of the frame's DA field are 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.

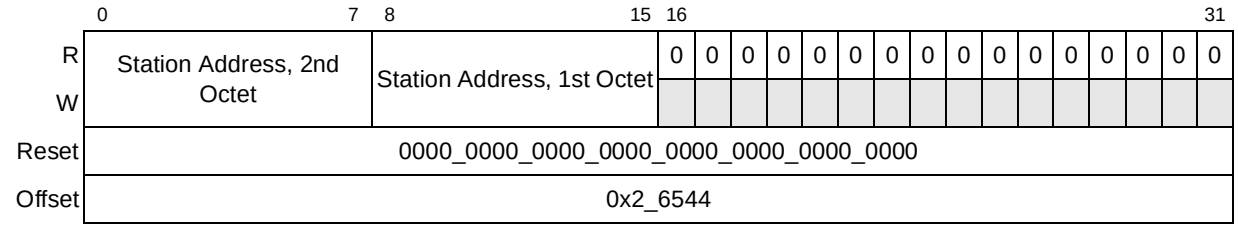


Figure 21-37. Station Address Part 2 Register Definition

Table 21-36 describes the fields of the MACSTNADDR2 register.

Table 21-36. MACSTNADDR2 Field Descriptions

Bit	Name	Description
0–7	Station Address, 2nd Octet	This field holds the second octet of the station address. The second octet (station address bits 8–15) defaults to a value of 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

21.5.3.7 Hash Function Registers

This section provides detailed descriptions of the registers used for hash functions. All of the registers are 32 bits wide. When the DA field of a receive frame is processed through a 32-bit CRC generator, the 8-bit MSB of the CRC remainder is mapped to a hash table entry. The user can enable a hash entry by setting the appropriate bit. A hash entry usually represents a set of addresses. A hash table hit occurs when the DA CRC result points to an enabled hash entry. The user must further filter the address. See Section 21.6.2.5.2, “Hash Table Algorithm,” for more information on the hash algorithm.

21.5.3.7.1 Individual Address Registers 0–7 (IADDR*n*)

The IADDR*n* registers, shown in Figure 21-38, 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 when the DA CRC result points to an enabled hash entry.

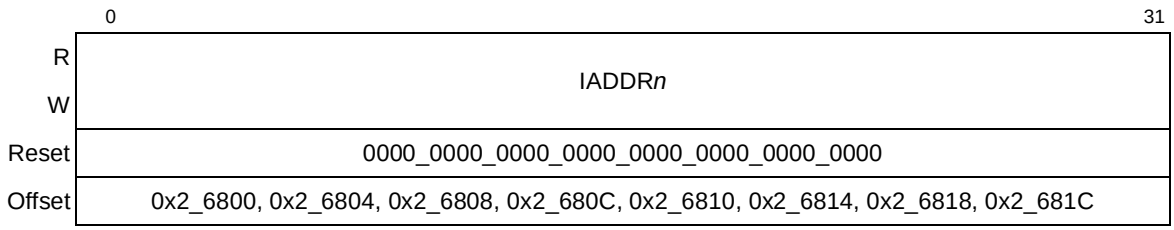


Figure 21-38. IADDR*n* Register Definition

Table 21-37 describes the field of the IADDR*n* registers.

Table 21-37. 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.

21.5.3.7.2 Group Address Registers 0–7 (GADDR*n*)

The GADDR*n* registers, shown in Figure 21-39, represent 256 entries of the group (multicast) address hash table used in the address recognition process. When the DA field of a receive frame is processed through a 32-bit CRC generator, the 8-bit MSB of the CRC remainder is mapped to one of the 256 entries. The user can enable a hash entry by setting the appropriate bit. A hash table hit occurs when the DA CRC result points to an enabled hash entry.

	0	15
R W	GADDR*n*	
Reset	0000_0000_0000_0000_0000_0000_0000_0000	
Offset	0x2_6880, 0x2_6884, 0x2_6888, 0x2_688C, 0x2_6890, 0x2_6894, 0x2_6898, 0x2_689C	

Figure 21-39. GADDR*n* Register Definition

Table 21-38 describes the field of the GADDR*n* registers.

Table 21-38. 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.

21.5.3.8 Attribute Registers

This section describes the two FEC frame attribute registers.

21.5.3.8.1 Attribute Register (ATTR)

The ATTR, shown in Figure 21-40, defines attributes and transaction types used to access buffer descriptors and 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. Extraction is specified in conjunction with ATTRELI. ATTR[ELCWT] only has meaning if the ATTRELI[EL] field is non-zero.

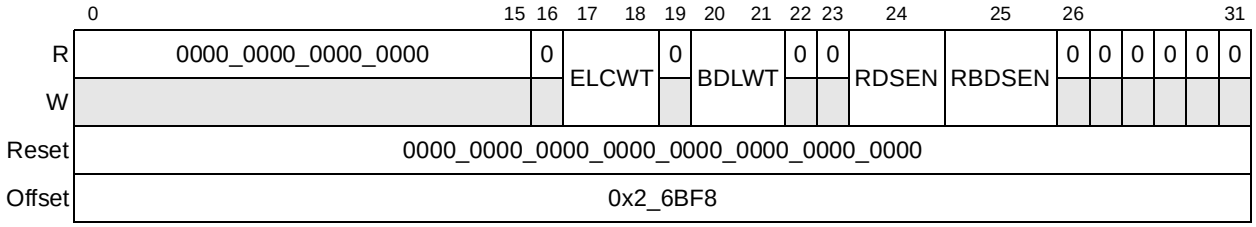


Figure 21-40. ATTR Register Definition

Table 21-39 describes the fields of the ATTR register.

Table 21-39. 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 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.
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 superceded 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

21.5.3.8.2 Attribute Extract Length and Extract Index Register (ATTRELI)

The ATTRELI registers are written by the user to specify the extract index and extract length. Figure 21-41 describes the definition for ATTRELI.

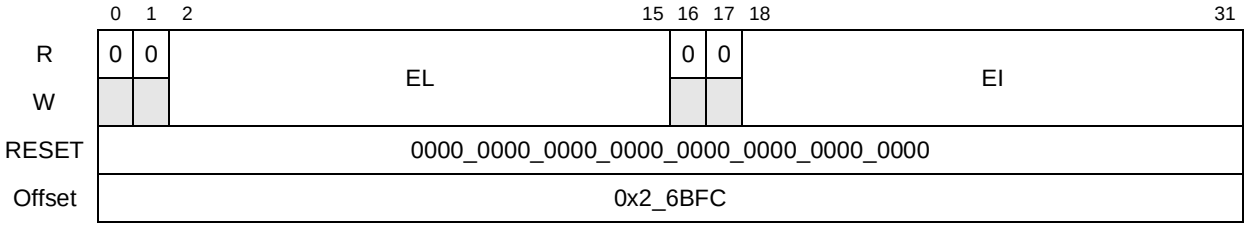


Figure 21-41. ATTRELI Register Definition

Table 21-40 describes the fields of the ATTRELI register.

Table 21-40. 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.

21.6 Functional Description

This section describes many of the functions of the 10/100 fast Ethernet controller (FEC).

21.6.1 Connecting to the Physical Interface

This section describes how to connect the FEC to the MII interface. The bus follows the bus conventions used in the IEEE 802.3 specification because the PHY follows the same conventions (that is, in the bus TXD[3:0], bit 3 is the MSB and bit 0 the LSB).

21.6.1.1 Media-Independent Interface (MII)

This section describes the media independent interface (MII) intended to be used between the PHYs and the FEC. Figure 21-42 depicts the basic components of the MII including the signals required to establish the FEC module connection with a PHY.

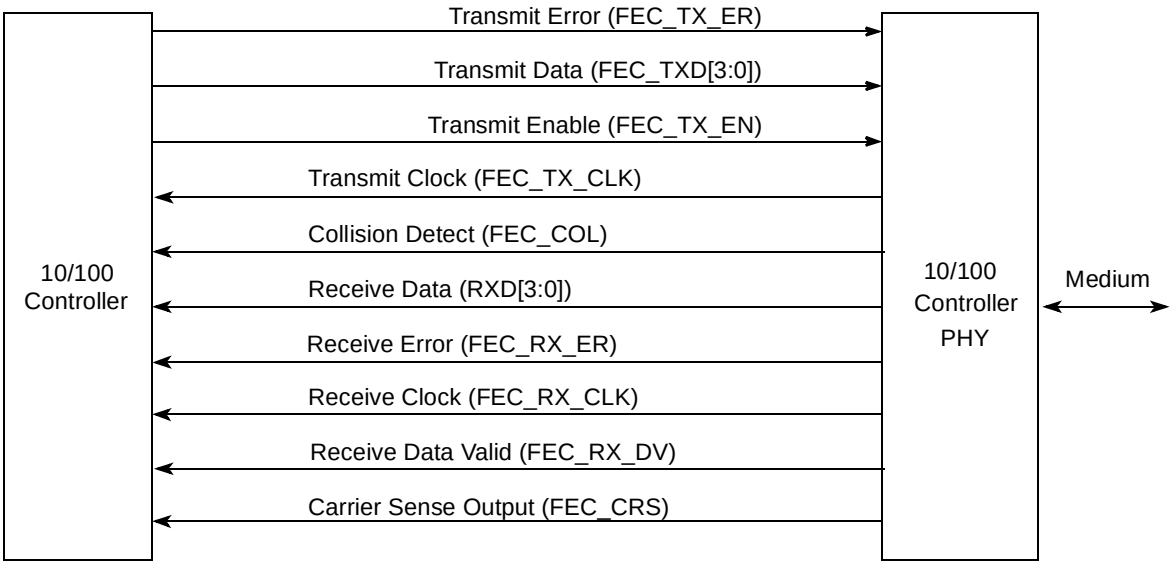


Figure 21-42. FEC–MII Connection

An MII interface has 16 signals, as defined by the IEEE 802.3u standard, for connecting to an Ethernet PHY. Table 21-41 lists the MII interface signals.

Table 21-41. MII Interface Signals

Frequency [MHz] 25 Voltage [V] 3.3					
Signals	**I/O**	**No. of Signals**	**Signals**	**I/O**	**No. of Signals**
FEC_TX_CLK	I	1	FEC_RxD[0]	I	1
FEC_TxD[0]	O	1	FEC_RxD[1]	I	1
FEC_TxD[1]	O	1	FEC_RxD[2]	I	1
FEC_TxD[2]	O	1	FEC_RxD[3]	I	1
FEC_TxD[3]	O	1	FEC_RX_DV	I	1
FEC_TX_EN	O	1	FEC_RX_ER	I	1
FEC_TX_ER	O	1	FEC_COL	I	1
FEC_RX_CLK	I	1	FEC_CRS	I	1

21.6.2 FEC Channel Operation

This section describes the operation of the FEC. First the software initialization sequence is described. Next, the software (Ethernet driver) interface for transmitting and receiving frames is described. Address recognition and hash table algorithm features are also discussed. The section concludes with interrupt handling, inter-packet gap time, and loop-back descriptions.

21.6.2.1 Initialization Sequence

This section describes which registers are reset due to a hard or software reset and what registers the user must initialize before the FEC is enabled.

21.6.2.1.1 Hardware Controlled Initialization

A hard reset occurs when the system powers up. All of the FEC's registers and control logic are reset to their default states after a hard reset has occurred.

21.6.2.1.2 User Initialization

After the system has undergone a hard reset, software must initialize certain basic FEC registers. Other registers can also be initialized during this time, but they are optional and must be determined based on the requirements of the system. See Table 21-2 for the register list. Table 21-42 describes the minimum steps for register initialization.

Table 21-42. Steps of Minimum Register Initialization

Description
1. Set, then clear MACCFG1 [Soft_Reset]
2. Initialize MACCFG2
3. Initialize MAC station address
4. Set up the PHY using the MII Mgmt Interface
5. Clear IEVENT
6. Initialize IMASK
7. Initialize IADDR*n*
8. Initialize GADDR*n*
9. Initialize RCTRL
10. Initialize DMACTRL

After the registers are initialized, the user must execute the following steps in the order described below to bring the FEC into a functional state (out of reset):

1. For the transmission of Ethernet frames, TxBDs have to be first built in memory, linked together as a ring, and pointed to by the TBASE register. At least two buffer descriptors per ring are required. Setting the ring size to one causes the same frame to be sent 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 the MACCFG1 register and set the appropriate bits. These must include RX_EN = 1, and TX_EN = 1. 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 was previously stopped. See Section 21.5.3.1.6, “DMA Control Register (DMACTRL),” and Section 21.6.3.1, “Transmit Data Buffer Descriptor (TxBD).”

21.6.2.2 Soft Reset and Reconfiguring Procedure

Before issuing a soft-reset to and/or reconfiguring the MAC with new parameters, the user must properly shutdown the DMA and make sure it is in an idle state for the entire duration. The 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.

21.6.2.3 FEC Frame Transmission

The Ethernet transmitter requires almost no core intervention. After the software driver initializes the system, the FEC begins to poll the first TxBD in the TxBD ring every 512 transmit clocks. If TxBD[R] is set, the FEC begins moving transmit buffer from memory to its Tx FIFO. The transmitter takes data from the Tx FIFO and transmits data to the MAC. The MAC transmits the data through the MII interface to the physical media. The transmitter, once initialized, runs until the end-of-frame (EOF) condition is detected unless a collision within the collision window occurs (half-duplex mode) or an abort condition is encountered.

If the user has a frame ready to transmit, a transmit-on-demand function may be emulated while in polling mode by using the graceful-transmit-stop feature. First, clear the IMASK[GTSCEN] bit to mask the graceful-transmit-stop complete interrupt. Next set, then immediately clear the DMACTRL[GTS] bit. Clear the resulting IEVENT[GTSC] bit. Finally, the IMASK[GTSCEN] bit may be set once again.

There is one internal buffer for out-of-sequence flow control frames. When the FEC is between frames, this buffer is polled. The buffer must contain the whole frame. Once the FEC is in paused mode, the out-of-sequence buffer descriptor cannot be used to send another flow control frame because the MAC treats it like a regular TxBD.

In half-duplex mode (MACCFG2[Full_Duplex] = 0) the MAC defers transmission if the line is busy (FEC_CRS asserted). Before transmitting, the MAC waits for carrier sense to become inactive, at which point it then determines if FEC_CRS remains negated for 60 clocks. If so, transmission begins after an additional 36 bit times (96 bit times after FEC_CRS originally became negated). If FEC_CRS continues to be asserted, the MAC follows a specified back-off procedure and tries to retransmit the frame until the retry limit is reached. Data stored in the Tx FIFO is retransmitted in case of a collision. This improves bus usage and latency.

The transmitter also monitors for an abort condition and terminates the current frame if an abort condition is encountered. In full-duplex mode the protocol is independent of network activity, and only the transmit inter-frame gap needs to be enforced.

The transmit block also implements full-duplex flow control. If a flow control frame is received, the MAC does not service the transmitter’s request to send data until the pause duration is over. If the MAC is currently sending data while a pause frame is received, the MAC 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

After the FCS is sent, the Ethernet controller writes the frame status bits into the BD and clears TxBD[R]. When the end of the current buffer is reached and TxBD[L] = 0 (a frame is comprised of multiple buffer descriptors), only TxBD[R] is cleared.

For both half- and full-duplex modes, an interrupt can be issued depending on TxBD[I]. The Ethernet controller then proceeds to the next TxBD in the table. In this way, the core can be interrupted after each frame, after each buffer, or after a specific buffer is sent. If TxBD[PAD/CRC] = 1, the Ethernet controller pads any frame shorter than 64 bytes.

To pause transmission, or rearrange the transmit queue, set DMACTRL[GTS]. This can be useful for transmitting expedited data ahead of previously linked buffers or for error situations. When this bit is set the FEC transmitter performs a graceful transmit stop. The Ethernet controller stops immediately if no transmission is in progress or continues transmission until the current frame either finishes or terminates with an error. The IEVENT[GTSC] interrupt occurs when the graceful transmit stop operation is completed. When DMACTRL[GTS] is cleared, the FEC resumes transmission with the next frame.

The FEC sends bytes least-significant nibble first and each nibble is sent least-significant bit first.

21.6.2.4 FEC Frame Reception

The FEC Ethernet receiver is designed to work with almost no 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 21.6.4, “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. The MAC checks for when FEC_RX_DV is asserted and as long as FEC_COL remains negated (full-duplex mode ignores FEC_COL) the MAC looks for the start of a frame by searching for a valid preamble/SFD (start of frame delimiter) header, which is stripped, and the frame begins to be processed. If a valid header is not found, the frame is ignored.

If the receiver detects the first bytes of a frame, the FEC begins to perform the frame recognition function through destination address (DA) recognition. (See Section 21.6.2.5, “Frame Recognition” for additional information.) Based on this match the frame can be accepted or rejected. Once accepted, the FEC processes the frame based on user-defined attributes.

The receiver can also filter frames based on physical (individual), group (multicast), and broadcast addresses. Because Ethernet receive frame data is not written to memory until the internal frame

recognition algorithm is complete, system bus usage is not wasted on frames not wanted by this station.

If a frame is accepted, the Ethernet controller fetches the receive buffer descriptor (RxBD) from the queue. If the RxBD is not being used by software (RxBD[E] is set), the FEC begins transferring the incoming frame. RxBD[F] is set for the first RxBD used for any receive frame.

If the buffer is filled, the FEC clears RxBD[E] and generates an interrupt if RxBD[I] is set. If the incoming frame is larger than the buffer, the Ethernet controller fetches the next RxBD in the table; if it is empty, it continues receiving the rest of the frame. In half-duplex mode, if a collision is detected during the frame, no RxBDs are used. Thus, no collision frames are presented to the user except late collisions, which indicate serious LAN problems.

The RxBD length is determined by the MRBL field in the maximum receive buffer length register (MRBLR). The smallest valid value is 64 bytes. During reception, the Ethernet controller checks for frames that are too short or too long. When the frame ends (FEC_CRS is negated), the receive CRC field is checked and written to the data buffer. The data length written to the last RxBD in the Ethernet frame is the length of the entire frame, which enables the software to recognize a frame-too-long condition.

Receive frames are not truncated if they exceed maximum frame bytes in the MAC's maximum frame register if MACCFG2[Huge Frame] is set, however the babbling receiver error interrupt occurs (IEVENT[BABR] is set) and RxBD[LG] is set.

When the receive frame is complete, the Ethernet controller sets RxBD[L], updates the frame status bits in the RxBD, and clears RxBD[E]. If RxBD[I] is set, the Ethernet controller next generates a maskable interrupt, indicating that a frame was received and is in memory. The Ethernet controller then waits for a new frame.

To interrupt reception, or rearrange the receive queue, DMACTRL[GRS] must be set. When this bit is set the FEC receiver does a graceful receive stop. The Ethernet controller stops immediately if no frames are being received or continues receiving until the current frame either finishes or an error condition occurs. The IEVENT[GRSC] interrupt event is signalled when the graceful receive stop operation is completed. While in this mode the user can then clear IEVENT[GRSC] and can write to registers that are accessible to both user and the FEC hardware without fear of conflict. When DMACTRL[GRS] is cleared, the FEC scans the input data stream for the start of a new frame (preamble sequence and start of frame delimiter), it resumes receiving, and the first valid frame received is placed in the next available RxBD.

21.6.2.5 Frame Recognition

The Ethernet controller performs frame recognition using destination address (DA) recognition. A frame can be rejected or accepted based on the outcome.

21.6.2.5.1 Destination Address Recognition

The Ethernet controller can also perform the frame filtering using the traditional destination address (DA) recognition methods. Figure 21-43 depicts a flowchart for address recognition on received frames that is used to explain the concept. In the actual implementation most of the decision points shown in the figure actually occur simultaneously.

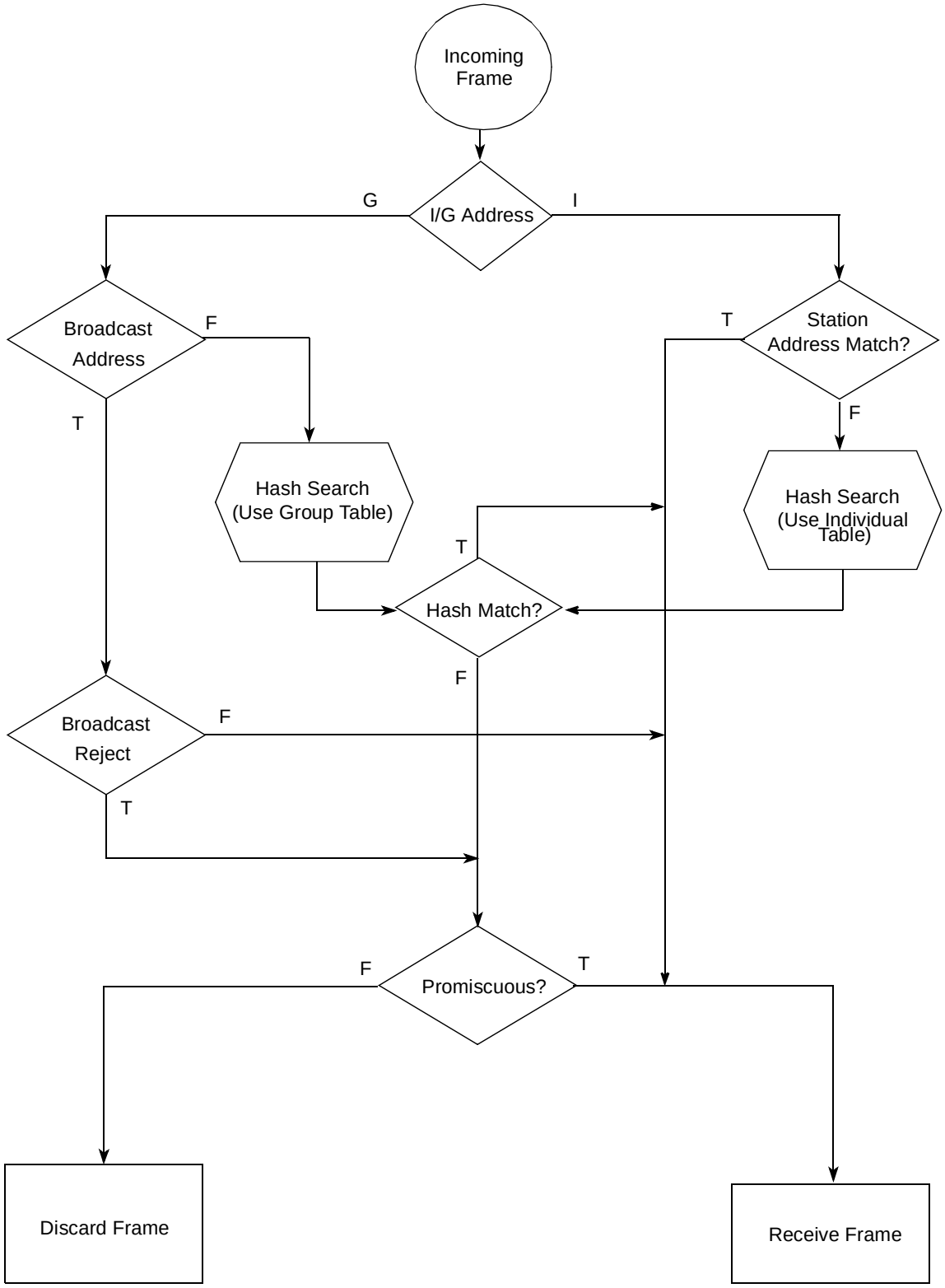


Figure 21-43. Ethernet Address Recognition Flowchart

The Ethernet controller compares the destination address field of the received frame with the physical address that the user programs in the station address registers (MACSTNADDR1 and MACSTNADDR2). If the DA does not match the station address, then the FEC performs address recognition on multiple individual addresses using the IADDR*n* hash table. The user must write zeros to the hash in order to avoid a hash match, and ones to station address in order to avoid individual address match, or the user can turn on promiscuous mode (See Section 21.5.3.4.1, “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.

21.6.2.5.2 Hash Table Algorithm

The hash table process used in the individual and group hash filtering operates as follows. The Ethernet controller maps any 48-bit destination address into one of 256 bins, represented by the 256 bits in GADDR0–7 or IADDR0–7. The eight high-order bits of a cyclic redundancy check (CRC) checksum are used to index into the hash table. The high-order three bits of this 8-bit field are used to select one of the eight registers in either the individual or group hash table. The low-order five bits select a bit within the 32-bit register. A value of 0 in the high-order three bits selects IADDR0/GADDR0.

The same process is used if the Ethernet controller receives a frame. If the CRC checksum selects a bit that is set in the group/individual hash table, the frame is accepted. If 32 group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 224/256 (87.5%) of the group address frames from reaching memory. Software must further filter those that reach memory to determine if they contain the correct addresses.

Better performance is achieved by using the group and individual hash tables in combination. For instance, if 32 group and 32 physical addresses are stored in their respective hash tables, because 87.5% of all group addresses and 87.5% of all individual address are rejected, then 87.5% of all frames are prevented from reaching memory.

The effectiveness of the hash table declines as the number of addresses increases. For instance, as the number of addresses stored in the 256-bin hash table increases, the vast majority of the hash table bits are set, preventing only a small fraction of frames from reaching memory.

21.6.2.5.3 CRC Computation Examples

There are many algorithms for calculating the CRC value of a number. Refer to the RFC 3309 standard, which can be found at http://www.faqs.org/rfcs/rfc3309.html, to compute the CRC value

for the purposes of FEC. The RFC 3309 algorithm uses the following polynomial to calculate the CRC value: x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x1+x0 or 0x04c11db7.

Given a destination MAC address of DA=01000CCCCCCC, the algorithm results in a CRC remainder value of 0xA29F4BBC.

Bit-reversing the low-order byte of the CRC value (0xBC) yields BR_CRC = 0x3D = 0b00111101

The high-order 3-bits of the new BR_CRC value are used to select which 32-bit register (of the 8) to use. This example maps the DA to register 1.

High-order 3 bits of BR_CRC: HO_CRC = 0b001 = 1

The low-order 5 bits are used to select which bit to set in the given register (with a value of 0 setting 0x8000_0000 and 31 setting 0x0000_0001). Therefore, the example DA maps to bit 29 of register 1.

Low-order 5 bits of BR_CRC: LO_CRC = 0b11101 = 29

Therefore, GADDR1 is ORed with the value 0x0000_0004.

Additional calculated examples follow:

Example 1:

- Destination MAC address: DA = 01005E000128
- CRC remainder value: CRC = 0x821D6CD3
- Bit-reversed least-significant byte of CRC value: BR_CRC = 0xCB = 0b11001011
- High-order 3 bits of BR_CRC: HO_CRC = 0b110 = 6
- Low-order 5 bits of BR_CRC: LO_CRC = 0b01011 = 11
- GADDR6 = 0x0010_0000

Example 2:

- Destination MAC address: DA = 0004F0604F10
- CRC remainder value: CRC = 0x1F5A66B5
- Bit-reversed least-significant byte of CRC value: BR_CRC = 0xAD = 0b10101101
- High-order 3 bits of BR_CRC: HO_CRC = 0b101 = 5
- Low-order 5 bits of BR_CRC: LO_CRC = 0b01101 = 13
- GADDR5 = 0x0004_0000

21.6.2.6 Flow Control

Because collisions cannot occur in full-duplex mode, the FEC can operate at the maximum rate. When the rate becomes too fast for a station's receiver, the station's transmitter can send flow-control frames to reduce the rate. Flow-control instructions are transferred by special frames of minimum frame size. The length/type fields of these frames have a special value. Table 21-43 shows the flow control frame structure.

Table 21-43. Flow Control Frame Structure

Size [Octets]	Description	Value	Comment
7	Preamble		
1	SFD		Start frame delimiter
6	Destination address	01-80C2-00-00-01	Multicast address reserved for use in MAC frames
6	Source address		
2	Length/type	88-08	Control frame type
2	MAC opcode	00-01	Pause command
2	MAC parameter		Pause period measured in slot times, most-significant octet first with a two time-slot resolution. Note: Because the pause period has a resolution of two time slots, the value programmed in this field is rounded up to the nearest even number before being used, as follows: **MAC Parameter Value** — **Pause Period** 0 — None 1 or 2 — 2 × slot time 3 or 4 — 4 × slot time … — …
42	Reserved	—	
4	FCS		Frame check sequence (CRC)

When 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.

21.6.2.7 Interrupt Handling

The following describes what usually occurs within a FEC interrupt handler:

- When an interrupt occurs, read IEVENT to determine interrupt sources. IEVENT bits to be handled in this interrupt handler are normally cleared at this time.

- Process the TxBDs to reuse them if the IEVENT[TXB] or IEVENT[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 FEC. Thus, it is important to check more than just one TxBD during the interrupt handler. One common practice is to process all TxBDs in the interrupt handler until one is found with R set. See Table 21-44.
- Obtain data from the RxBD if IEVENT[RXC], IEVENT[RXB], or IEVENT[RXF] is set. If the receive speed is fast or the interrupt delay is long, the FEC 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 bit E set. Because the FEC pre-fetches BDs, the BD table must be big enough so that there is always another empty BD to pre-fetch. See Table 21-45.
- Clear any set halt bits in TSTAT and RSTAT registers, or DMACTRL[GTS] and DMACTRL[GRS].
- Continue normal execution.

Table 21-44. Non-Error Transmit Interrupts

Interrupt	Description	Action taken by FEC
GTSC	Graceful transmit stop complete: Transmitter is put into a pause state after completion of the frame currently being transmitted.	None
TXC	Transmit control: Instead of the next transmit frame, a control frame was sent.	None
TXB	Transmit buffer: A TxBD, 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 TxBD of that frame was updated.	Programmable write-with-response to memory on the last TxBD before setting IEVENT[TXF].

Table 21-45. Non-Error Receive Interrupts

Interrupt	Description	Action taken by FEC
GRSC	Graceful receive stop complete: Receiver is put into a pause state after completion of the frame currently being received.	None
RXC	Receive control: A control frame was received. As soon as the transmitter finishes sending the current frame, a pause operation is performed lasting for the duration specified in the received pause control frame and beginning when the frame was first received.	None
RXB	Receive buffer: An RxBD, 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 RxBD of that frame was updated.	Programmable write-with-response to memory on the last RxBD before setting IEVENT[RXF].

21.6.2.8 Inter-Packet Gap Time

When a station needs to transmit, it waits until the LAN becomes silent for a specified period (inter-packet gap). When a station starts sending, it continually checks for collisions on the LAN. If a collision is detected, the station forces a jam signal (all ones) on its frame and stops transmitting. Collisions usually occur close to the beginning of a packet. The station then waits a random time period (back-off) before attempting to send again. When 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 specified number of retries, an error is indicated.

The minimum inter-packet gap time for back-to-back transmission is 96 serial clocks. The receiver receives back-to-back packets with this minimum spacing. In addition, after waiting a required number of clocks (based on the back-off algorithm), the transmitter waits for carrier sense to be negated before retransmitting the packet. Retransmission begins 36 serial clocks after carrier sense is negated for at least 60 serial clocks.

21.6.2.9 Internal and External Loop Back

Setting MACCFG1[Loop_Back] causes MAC transmit outputs to be looped back to the MAC receive inputs. Clearing it results in normal operation. This bit is cleared by default.

21.6.2.10 Error-Handling Procedure

The Ethernet controller reports frame reception and transmission error conditions using the channel BDs, the error counters, and the IEVENT register.

Transmission errors are described in Table 21-46.

Table 21-46. Transmission Errors

Error	Response
Transmitter underrun	The FEC 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 FEC 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).
Late collision	The FEC 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 FEC 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.

Table 21-46. Transmission Errors (continued)

Error	Response
Babbling transmit error	A frame is transmitted which exceeds the MAC's Maximum Frame Length and MACCFG2[Huge Frame] is a 0. The FEC 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 21-47.

Table 21-47. Reception Errors

Error	Description
Overrun error	The FEC 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 FEC sets IEVENT[BSY]. In addition, RSTAT[RHLT] is set. The halted queue resumes reception once the user clears RSTAT[RHLT].
Non-octet error (dribbling bits)	The Ethernet controller handles a nibble of dribbling bits if the receive frame terminates as non-octet aligned and it checks the CRC of the frame on the last octet boundary. If there is a CRC error, the frame non-octet aligned (RxBD[NO]) error is reported, IEVENT[RXF] is set, and the alignment error counter increments. The FEC relies on the statistics collector block to increment the receive alignment error counter (RALN). If there is no CRC error, no error is reported.
CRC error	If a CRC error occurs, the FEC sets RxBD[CR], closes the buffer, and sets IEVENT[RXF]. This FEC relies on the statistics collector block to record the event. After receiving a frame with a CRC error, the receiver then enters hunt mode.
Memory Read Error	A system bus error occurred during a DMA transaction. The FEC sets IEVENT[EBERR] and discards the frame. Also, RSTAT[RHLT] is set. The halted queue resumes reception once RSTAT[RHLT] is cleared.
Babbling Receive Error	A frame is received that exceeds the MAC's maximum frame length. The FEC sets IEVENT[BABR] and continues.

21.6.3 Buffer Descriptors

The FEC 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 FEC descriptor base registers point to the beginning of BD rings.

There is an 8-byte data BD format designed to be similar to the existing MPC8260 BD model.

The FEC is capable of placing extracted data directly into the L2 cache memory. This allows the processor to quickly access critical frame information as soon as the processor is ready without having to first fetch the data from main memory. This results in substantial improvement in throughput and hence reduction in latency.

Data buffers are used in the transmission and reception of Ethernet frames (see Figure 21-44). Data BDs encapsulate all information necessary for the FEC to transmit or receive an Ethernet frame. Within each data BD there is a status field, a data length field, and a data pointer. The BD completely describes an Ethernet packet by centralizing status information for the data packet in the status field of the BD and by containing a data BD pointer to the location of the data buffer. Software is responsible for setting up the BDs in memory. Because of prefetching, at least 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. An ownership bit in the status field defines the current state of the buffer (pointed to by the data pointer). Other status field bits

in the buffer descriptor communicate status/control information between the FEC and the software driver.

Because there is no next BD pointer in the transmit/receive BD (see Figure 21-45), all BDs must reside sequentially in memory. The FEC increments the current BD location appropriately to the next BD location to be processed. There is a wrap bit in the last BD that informs the FEC to loop back to the beginning of the BD chain. Software must initialize the TBASE and RBASE registers that point to the beginning transmit and receive BDs for the FEC.

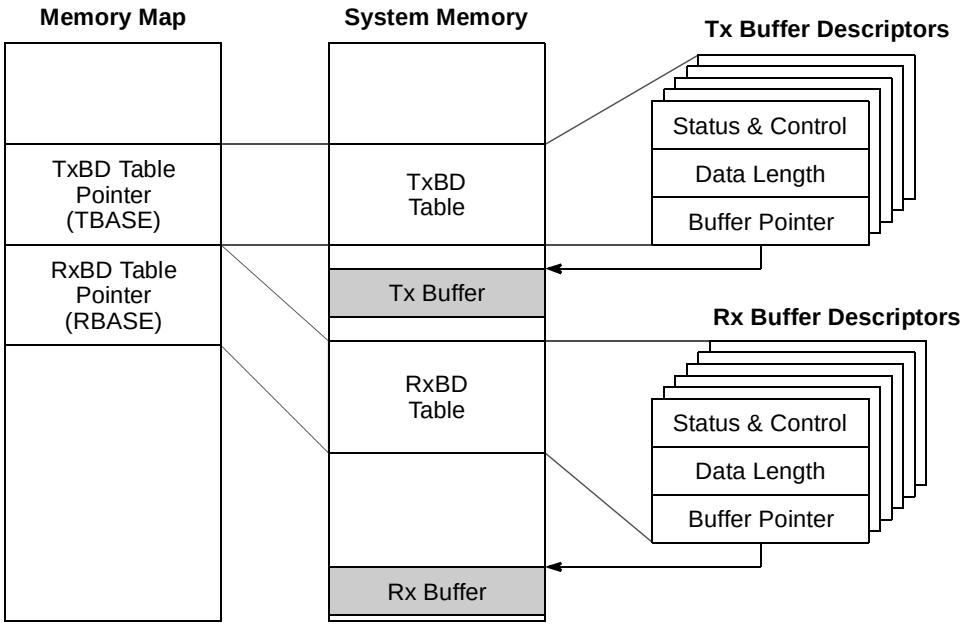


Figure 21-44. Example of FEC Memory Structure for BD

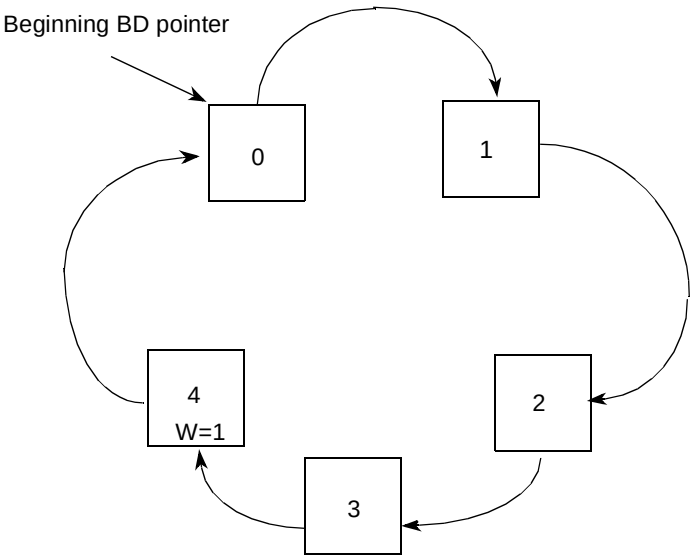


Figure 21-45. Buffer Descriptor Ring

21.6.3.1 Transmit Data Buffer Descriptor (TxBD)

Data is presented to the FEC for transmission by arranging it in memory buffers referenced by the TxBDs. In the TxBD the user initializes the R, PAD/CRC, W, L, and TC bits and the length (in bytes) in the first word, and the buffer pointer in the second word.

The FEC clears the R bit in the first word of the BD after it finishes using the data buffer. The transfer status bits are then updated. Additional transmit frame status can be found in statistic counters in the MIB block. Figure 21-46 describes the TxBD.

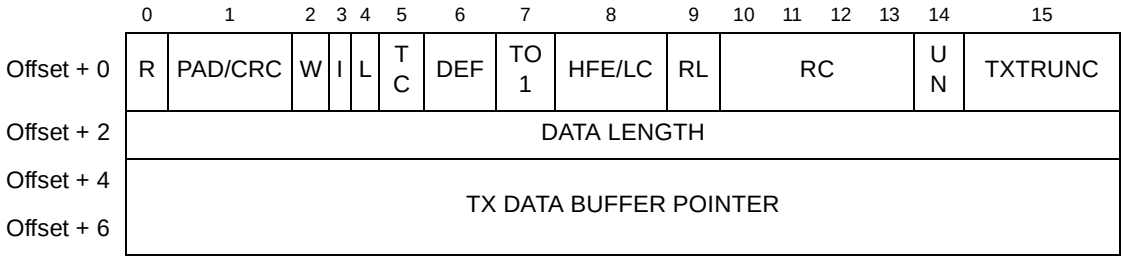


Figure 21-46. Transmit Buffer Descriptor

The TxBD fields are detailed in Table 21-48.

Table 21-48. Transmit Data Buffer Descriptor (TxBD) Field Descriptions

Offset	Bits	Name	Description
Offset + 0	0	R	Ready. Written by the FEC and user. 0 The data buffer associated with this BD is not ready for transmission. The user is free to manipulate this BD or its associated data buffer. The FEC clears this bit after the buffer is transmitted or after an error condition is encountered. 1 The data buffer, which is prepared for transmission by the user, was not transmitted or is currently being transmitted. No fields of this BD may be written by the user once this bit is set.
Offset + 0	1	PAD/CRC	PAD/CRC. Padding and CRC attachment for frames. (Valid only while it is set in the first BD and MACCFG2[PAD/CRC] is cleared.) If MACCFG2[PAD/CRC] is set, this bit is ignored. 0 Do not add padding to short frames. No CRC is appended unless TxBD[TC] is set. 1 Add PAD/CRCs to frames. Pad bytes are inserted until the length of the transmitted frame equals 64 bytes. Unlike the MPC8260 which pads up to MINFLR value, the FEC pads always up to the IEEE minimum frame length of 64 bytes. CRC is always appended to frames.
Offset + 0	2	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in TBASE.
Offset + 0	3	I	Interrupt. Written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[TXB] or IEVENT[TXF] are set after this buffer is serviced. These bits can cause an interrupt if they are enabled (That is, IEVENT[TXBEN] or IEVENT[TXFEN] are set).

Table 21-48. Transmit Data Buffer Descriptor (TxBD) Field Descriptions (continued)

Offset	Bits	Name	Description
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.
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 the FEC) Huge frame enable. Written by user. Valid only while it is set in first BD and the MACCFG2[Huge Frame] is cleared. If MACCFG2[Huge Frame] is set, this bit is ignored. 0 Truncate transmit frame if its length is greater than the MAC's Maximum Frame Length register. 1 Do not truncate the transmit frame. Late collision. Written by the FEC. 0 No late collision. 1 A collision occurred after 64 bytes are sent. The FEC terminates the transmission and updates LC.
Offset + 0	9	RL	Retransmission Limit. Written by the FEC. 0 Transmission before maximum retry limit is hit. 1 The transmitter failed (maximum retry limit + 1) attempts to successfully send a message due to repeated collisions. The FEC terminates the transmission and updates RL.
Offset + 0	10–13	RC	Retry Count. Written by the FEC. 0 The frame is sent correctly the first time or if RL is set then the retry limit has been reached x One or more attempts where needed to send the transmit frame. If this field is 15, then 15 or more retries were needed. The Ethernet controller updates RC after sending the buffer.
Offset + 0	14	UN	Underrun. Written by the FEC. 0 No underrun encountered (data was retrieved from external memory in time to send a complete frame). 1 The Ethernet controller encountered a transmitter underrun condition while sending the associated buffer. The FEC terminates the transmission and updates UN.

Table 21-48. Transmit Data Buffer Descriptor (TxBD) Field Descriptions (continued)

Offset	Bits	Name	Description
Offset + 0	15	TXTRUNC	TX truncation. Set in the last TxBD (TxBD[L] is set) when IEVENT[BABT] occurs for the frame.
Offset + 2	0–15	Data Length	Data length is the number of octets the FEC transmits from this BD's data buffer. It is never modified by the FEC. This field must be greater than zero. This field must be greater than zero.
Offset + 4	0–31	TX Data Buffer Pointer	The transmit buffer pointer contains the address of the associated data buffer. There are no alignment requirements for this address.

21.6.3.2 Receive Buffer Descriptor (RxBD)

In the RxBD the user initializes the E, I, and W bits in the first word and the pointer in second word. If the data buffer is used, the FEC modifies the E, L, F, M, BC, MC, LG, NO, SH, CR, OV, and TR bits and writes the length of the used portion of the buffer in the first word. The M, BC, MC, LG, NO, SH, CR, OV, and TR bits in the first word of the buffer descriptor are modified by the FEC only if the L bit is set. The first word of the RxBD contains control and status bits. Figure 21-47 describes the RxBD.

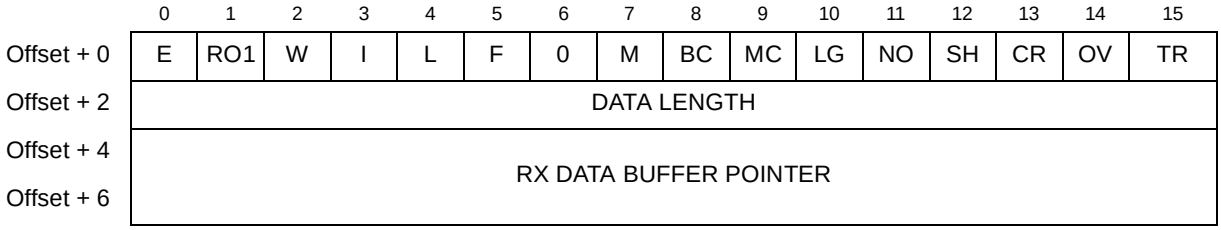


Figure 21-47. Receive Buffer Descriptor

Table 21-49 describes the fields of the RxBD.

Table 21-49. Receive Buffer Descriptor Field Descriptions

Offset	Bits	Name	Description
Offset + 0	0	E	Empty. Written by the FEC (when cleared) and by the user (when set). 0 The data buffer associated with this BD is filled with received data, or data reception is aborted due to an error condition. Status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	1	RO1	Receive software ownership bit. This field is reserved for use by software. This read/write bit is not modified by hardware, nor does its value affect hardware.
Offset + 0	2	W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in RBASE.

Table 21-49. Receive Buffer Descriptor Field Descriptions (continued)

Offset	Bits	Name	Description
Offset + 0	3	I	Interrupt. Written by user. 0 No interrupt is generated after this buffer is serviced. 1 IEVENT[RXB] or IEVENT[RXF] are set after this buffer is serviced. This bit can cause an interrupt if enabled (IMASK[RXBEN] or IMASK[RXFEN]). If the user wants to be interrupted only 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 the FEC. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 0	5	F	First in frame. Written by the FEC. 0 The buffer is not the first in a frame. 1 The buffer is the first in a frame.
Offset + 0	6	—	Reserved
Offset + 0	7	M	Miss. Written by the FEC. (This bit is valid only if L is set and the FEC is in promiscuous mode.) This bit is set by the FEC for frames that were accepted in promiscuous mode, but were flagged as a ‘miss’ by the internal address recognition; thus, while in promiscuous mode, the user can use bit M to quickly determine whether the frame was destined to this station. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode.
Offset + 0	8	BC	Broadcast. Written by the FEC. (Only valid if L is set.) Is set if the DA is broadcast (FF-FF-FF-FF-FF-FF).
Offset + 0	9	MC	Multicast. Written by the FEC. (Only valid if L is set.) Is set if the DA is multicast and not BC.
Offset + 0	10	LG	Rx frame length violation. Written by the FEC (only valid if L is set). A frame length greater than maximum frame length was recognized while MACCFG2[Huge Frame] was set. Note, if MACCFG2[Huge Frame] is cleared, the frame is truncated to the value programmed in the maximum frame length register.
Offset + 0	11	NO	Rx non–octet-aligned frame. Written by the FEC (only valid if L is set). A frame that contained a number of bits not divisible by eight was received.
Offset + 0	12	SH	Short frame. Written by the FEC (only valid if L is set). A frame length that was less than the minimum length defined for this channel (MINFLR) was recognized, provided RCTRL[RSF] is set.
Offset + 0	13	CR	Rx CRC error. Written by the FEC (only valid if L is set). This frame contains a CRC error and is an integral number of octets in length.This bit is also set if a receive code group error is detected.
Offset + 0	14	OV	Overrun. Written by the FEC (only valid if L is set). A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, SH, CR, and CL lose their normal meaning and are zero.
Offset + 0	15	TR	Truncation. Written by the FEC (only valid if L is set). Is set if the receive frame is truncated. This can happen if a frame length greater than maximum frame length was received and the MACCFG2[Huge Frame] is cleared. If this bit is set, the frame must be discarded and the other error bits must be ignored as they may be incorrect.

Table 21-49. Receive Buffer Descriptor Field Descriptions (continued)

Offset	Bits	Name	Description
Offset + 2	0–15	Data Length	Data length. Written by the FEC. Data length is the number of octets written by the FEC into this BD's data buffer if L is cleared (the value is equal to MRBL), or the length of the frame including CRC, if L is set.
Offset + 4	0–31	Rx Data Buffer Pointer	Receive buffer pointer. Written by user. The receive buffer pointer, which always points to the first location of the associated data buffer, must be 64-byte aligned. The buffer must reside in memory external to the FEC.

21.6.4 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 21.5.3.8.1, "Attribute Register (ATTR)," and Section 21.5.3.8.2, "Attribute Extract Length and Extract Index Register (ATTRELI)," for specific register information.

21.7 Application Information

21.7.1 Interface Mode Configuration

This section describes how to configure the FEC in the MII mode. The pinout, the data registers that must be initialized, as well as speed selection options are described.

21.7.1.1 MII Interface Mode

Table 21-50 shows the signal configuration needed for MII interface mode

Table 21-50. MII Interface Mode Signal Configuration

MII INTERFACE Frequency [MHz] 25 Voltage[V] 3.3		
Signals	**I/O**	**# of signals**
FEC_TX_CLK	I	1
FEC_TxD[0]	O	1
FEC_TxD[1]	O	1
FEC_TxD[2]	O	1
FEC_TxD[3]	O	1
FEC_TX_EN	O	1
FEC_TX_ER	O	1
FEC_RX_CLK	I	1
FEC_RxD[0]	I	1
FEC_RxD[1]	I	1
FEC_RxD[2]	I	1
FEC_RxD[3]	I	1
FEC_RX_DV	I	1
FEC_RX_ER	I	1
FEC_COL	I	1
FEC_CRS	I	1
sum		16

Table 21-51 describes the register initializations required to reconfigure the PHY to MII mode following intial auto-negotiation.

Table 21-51. MII Mode Register Initialization Steps

Set Soft_Reset, MACCFG1[1000_0000_0000_0000_0000_0000_0000_0000]
Clear Soft_Reset, MACCFG1[0000_0000_0000_0000_0000_0000_0000_0000]
Initialize MACCFG2, for MII, half duplex operation. Set I/F Mode bit to nibble mode, MACCFG2[0000_0000_0000_0000_0111_0001_0000_0100] (This example has Full Duplex = 0, Preamble count = 7, PAD/CRC append = 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.)
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]

Appendix A
Revision History

This appendix provides a list of the major differences between revisions of the *MPC8540 PowerQUICC III Integrated Host Processor Reference Manual, Revision 0* and *MPC8540 PowerQUICC III Integrated Host Processor Reference Manual, Revision 1*.

A.1 Changes From Revision 0 to Revision 1

Major changes to the *MPC8540 PowerQUICC III Integrated Host Processor Reference Manual* between Revision 0 to Revision 1 are as follows:

Section, Page	Changes
Chapters 4, 16, 18, 20	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, 2-18	In Table 2-9, correct the reset value for the I2CDFSRR register to be 0x10.
2.4, 2-27	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	14.5.3.2.1/14-30
0x2_4050–0x2_4088	Reserved	R	0x0000_0000	—

2.4, 2-27	In Table 2-9, correct the reset value for the OSTBD register to be 0x0800_0000.
2.4, 2-28	In Table 2-9, correct the MACCFG1 register from having R/W, R access to R/W access.
2.4, 2-28	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, add the following two rows under “FIFO Control and Status Registers”:

0x2_604C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	21.5.3.2.1/21-22
0x2_6050–0x2_6088	Reserved	R	0x0000_0000	—

2.4, 2-35 — In Table 2-9, correct the ATTR and ATTRELI registers from having R access to R/W access.

2.4, 2-35 — In Table 2-9, correct the offset for the Reserved addresses after the EOI register to be 0x4_00C0–0x4_0FF0.

2.4, 2-47 — In Table 2-9, correct the reset value for the GPINDR register to be 0x*nnnn*_0000.

2.4, 2-48 — In Table 2-9, correct the name of the register with offset 0xE_1094 from PMLCB5 to PMLCB8.

2.4, 2-42–46 — In Table 2-9, correct the access designation for all RapidIO Registers listed as “Special” to read “R/W”.

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.”

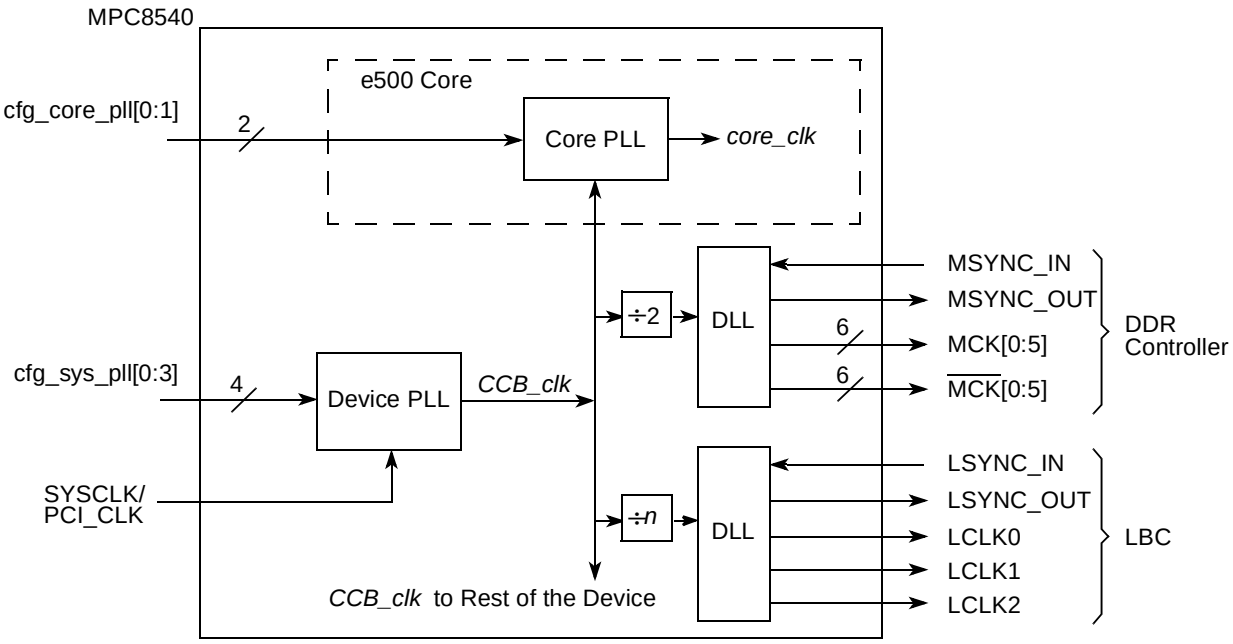
4.4.3.3, 4-14 — Add the following sentence to the end of the last paragraph:

“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, 5-18 — Remove the registers IAC3 and IAC4 from Figure 5-6.

5.14, 5-33 — 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. 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 16.3.1.4, “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 13.3.1.11, “Transfer Error Check Disable Register (LTEDR),” and Section 13.3.1.12, “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 17.3.2.1.2, “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 — IAC3 and IAC4 are not supported and should be removed from Figure 6-1.

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, 6-15 — 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, 6-27 — 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, 6-30	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, 6-37	In Table 6-28, describe bits 52–56 as “Reserved for implementation-specific use”.
6.13.3, 6-45	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, 7-28

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, 7-32

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."

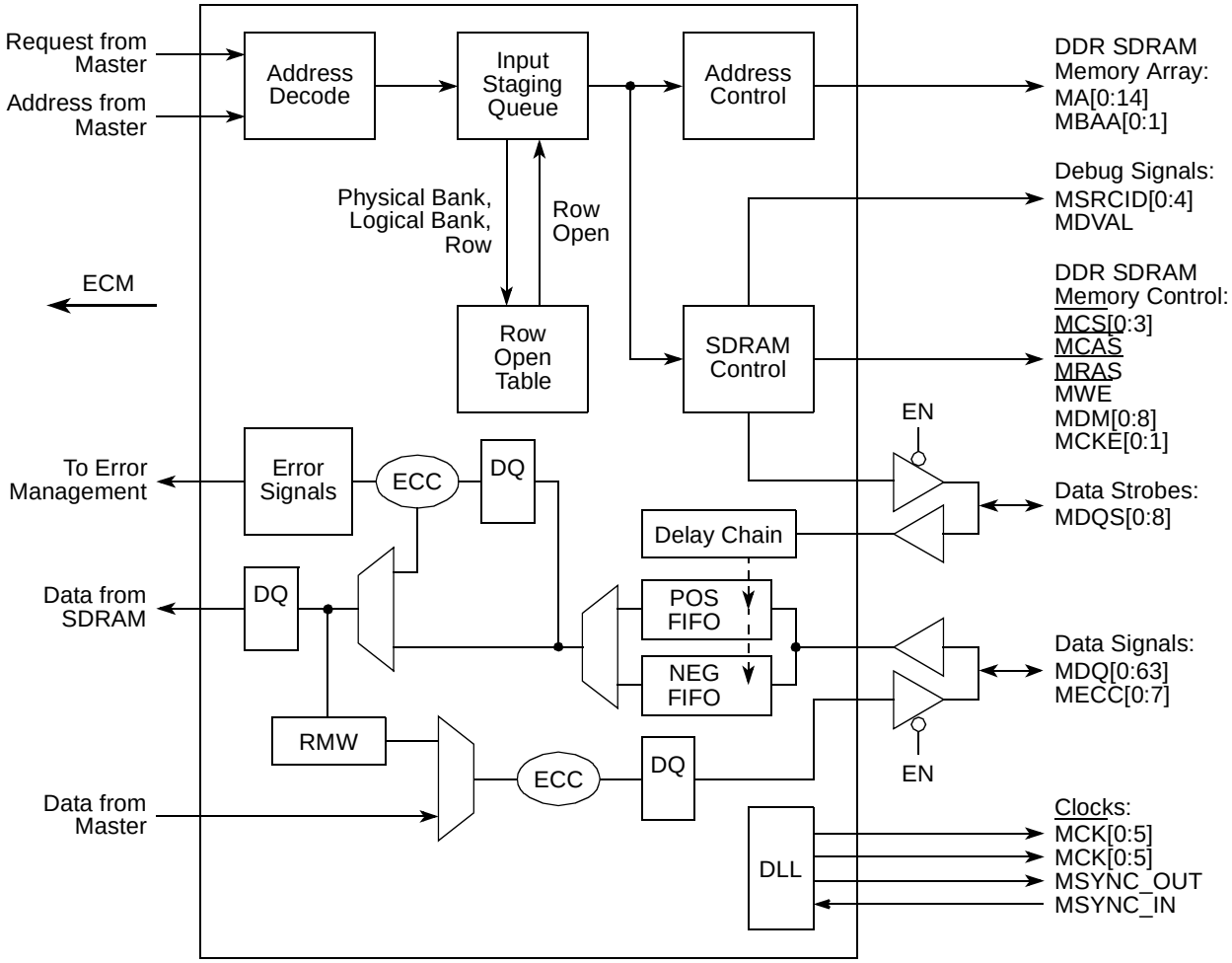
9.4.1.1, 9-10	In Table 9-6, change the CS*n*_BNDS[EA*n*] field description to read: "Ending address for chip select (bank) *n*. This value is compared against the 8 msbs of the address."
9.4.1.4, 9-13	In Table 9-9, replace all instances of |CASLAT| with ⌈CASLAT⌉ .
9.4.1.5, 9-14	In Table 9-10, change 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 18.5.1.5.3, "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

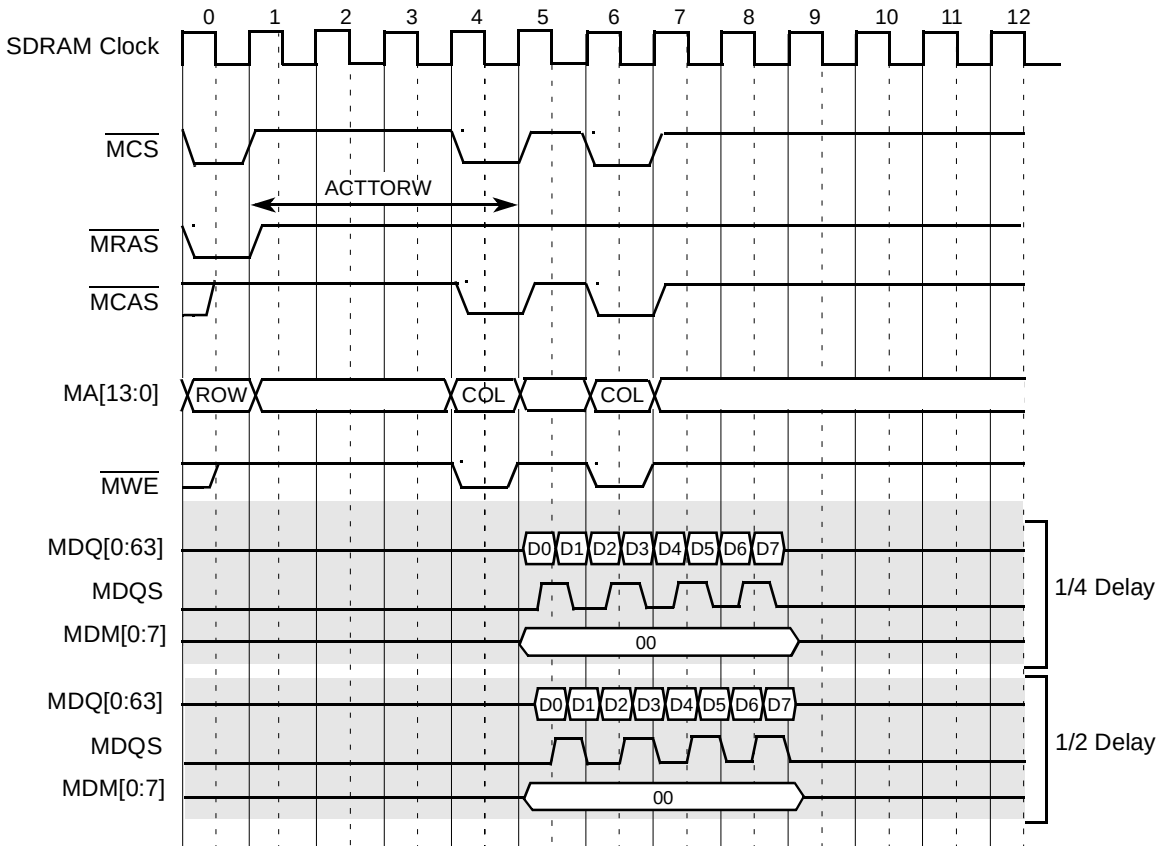
In all burst examples, re-number the data beats in the figures to be D0, D1, D2, D3, D0, D1, D2, D3.

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:

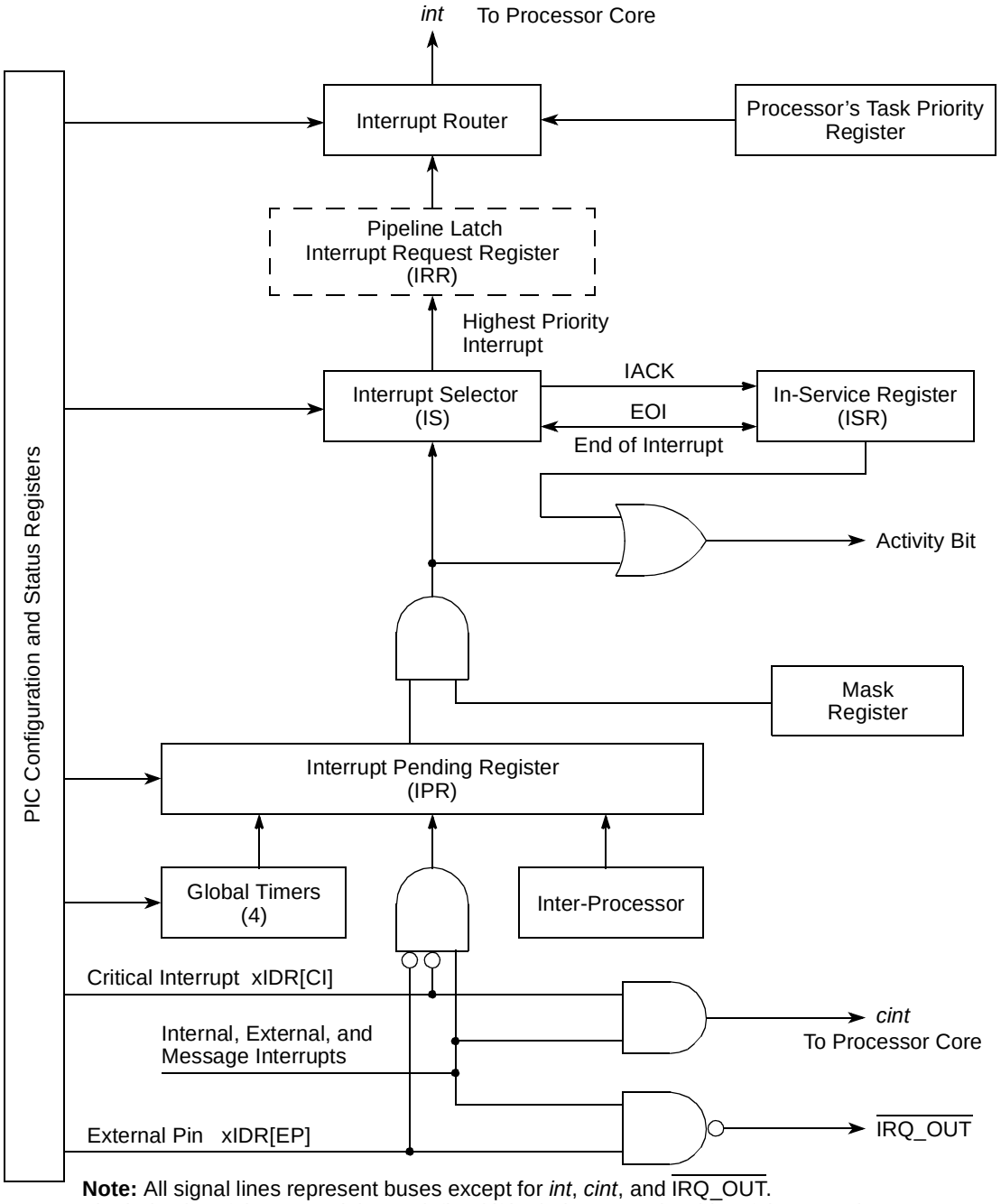


9.5.8, 9-40 In the paragraph preceding Section 9.5.8.1, change the reference to “Two sets of auto refresh commands...” to read “Three sets of auto refresh commands...”. Also, change the reference to “...commands are also staggered in two groups...” to read “... commands are also staggered in three groups...”

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, 10-7	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, 10-29	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, 10-36	In Table 10-35, correct the who am I register name from "WHOAMI0" to "WHOAMI."
10.4.1, 10-41	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, 10-41 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."
12.3.1.9, 12-15	In Table 12-16, change the second sentence of the asserted ("1") description for the ULSR[BI] to read: "A break condition is expected to last at least two character lengths and a new character is not loaded until SIN returns to the mark state (logic 1) and a valid START is detected."
Chapter 13	Correct all instances of "UPWAIT" to "LUPWAIT."
13.1.3.1, 13-3	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."
13.1.3.1, 13-4	In the last sentence, change "LCLK[0:3]" to "LCLK[0:2]."
13.2, 13-5	Delete the Reset State (outputs) column (and references to it) of Table 13-1.
13.2, 13-6, 13-7, and 13-9	In Table 13-2, change all instances of "RAS" to "$\overline{\text{RAS}}$" and all instances of CAS to $\overline{\text{CAS}}$.
13.3.1.2.3, 13-16 and 17	In Table 13-7, Table 13-8, Figure 13-4, and Figure 13-5, change bits 17–18 to reserved.
13.4.4, 13-61	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."

13.4.4.2, 13-65 — 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."

13.4.7.3, 13-55 — In the section title, change "CAS" to "$\overline{\text{CAS}}$".

Chapter 14, 21 — For consistency, change all instances of "HFE", "Huge_Frame", and "HUGE FRAME" to "Huge Frame".

14.2, 14-7 — In the first bullet item of the TSEC features list, change "820.3x" to "802.3x."

14.2, 14-7 — Change the fifth element of the second bullet item in the TSEC features list to read:

"– 10/100 Mbps RMII"

14.5.2, 14-14 — In Table 14-3, 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	14.5.3.2.1/A-19

14.5.2, 14-15 — In Table 14-3, correct the reset value for the OSTBD register to 0x0800_0000.

14.5.2, 14-16 — In Table 14-3, correct the MACCFG1 register from having R/W, R access to R/W access.

14.5.2, 14-16 — In Table 14-3, correct the IFSTAT register from having R access to R/W access.

14.5.2, 14-16 — In Table 14-3, change the reset value for the TSEC IFSTAT register to be 0x0000_0000.

14.5.2, 14-18 — In Table 14-3, correct the CAR1 and CAR2 registers from having R access to R/W access.

14.5.2, 14-18 — In Table 14-3, remove the incorrect "Cleared on read" footnote designation from the CAR1 and CAR2 registers.

14.5.2, 14-19 — In Table 14-3, correct the ATTR and ATTRELI registers from having R access to R/W access.

14.5.3.1.1, 14-20 — In Table 14-4, 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”

14.5.3.1.1, 14-20 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.”

14.5.3.1.1, 14-21 In Table 14-4, 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.”

14.5.3.1.2, 14-22 Replace the last sentence in the first paragraph with:

“The interrupt signal can be cleared by clearing the corresponding IEVENT bit.”

14.5.3.1.3, 14-24 Undocument bitfields TXEDIS, CRL/XDADIS, and XFUNDIS.

14.5.3.1.7, 14-27 and 28

Remove bitfield DMACTRL[TOD] from Figure 14-12 and Table 14-10. This bit position (29) is now reserved.

14.5.3.1.7, 14-28 In Table 14-10, 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.”

14.5.3.1.7, 14-28 In Table 14-10, 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.”

14.5.3.2.1, 14-30 Insert the following section as the first subsection of Section 14.5.3.2:

14.5.3.2.1 FIFO Pause Control Register (FIFO_PAUSE_CTRL)

FIFO_PAUSE_CTRL, shown in Figure 14-20, is writable by the user to configure the properties of the TSEC FIFO.

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

	16													29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TFC_PAUSE_EN	0
W																
Reset	0000_0000_0000_0000															
Offset	TSEC1:0x2_404C; TSEC2:0x2_504C															

Figure 14-14 FIFO_PAUSE_CTRL Register Definition

Table 14-20 describes the fields of the FIFO_PAUSE_CTRL register.

Table 14-12 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

14.5.3.3.1, 14-32 In Table 14-15, 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."

14.5.3.3.6, 14-35 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]."

14.5.3.3.8, 14-38 In Table 14-22, add the following sentence to the OSTBDLEN field description:

"This field must be greater than zero in order for a transfer to take place."

14.5.3.4.2, 14-40 In Table 14-25, 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.

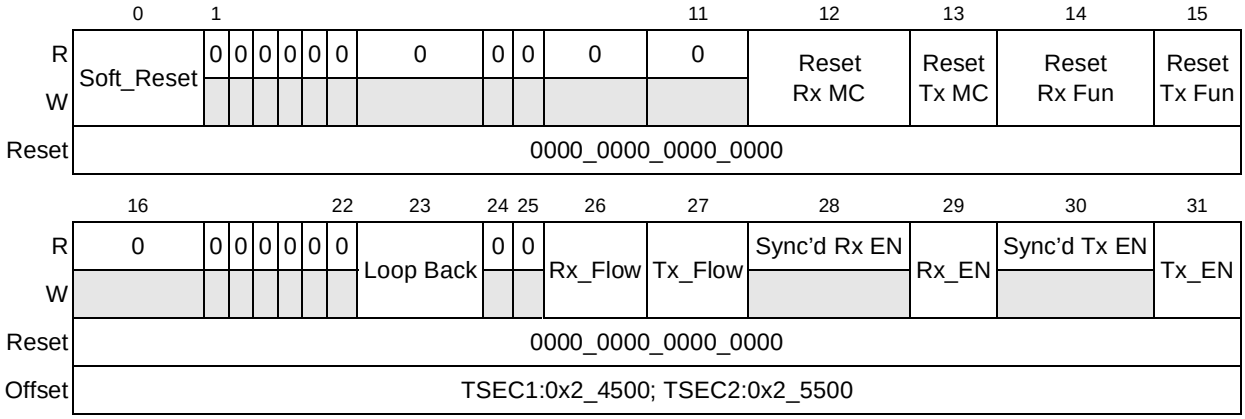
0 RxBD queue is enabled for Ethernet reception. (That is, it is not halted.)

1 All Ethernet controller receive activity to RxBD queue is halted."

14.5.3.4.2, 14-39 Replace Figure 14-27 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
W																														
Reset	0000_0000_0000_0000_0000_0000_0000_0000																													
Offset	TSEC1:0x2_4304; TSEC2:0x2_5304																													

14.5.3.6.1, 14-47 Replace Figure 14-34 with the following figure:



14.5.3.6.1, 14-48 In Table 14-32, add the following sentence to the MACCFG1[Rx_EN] field description:

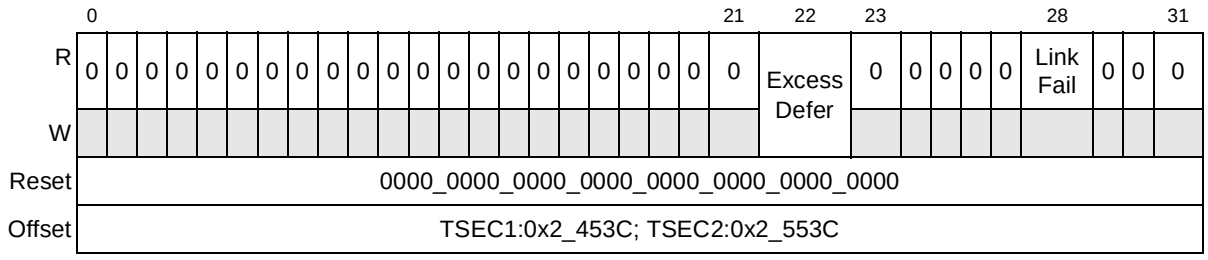
"If set, prior to clearing this bit, set DMACTRL[GRS] then confirm subsequent occurrence of the graceful receive stop interrupt (IEVENT[GRSC] is set)."

14.5.3.6.1, 14-48 In Table 14-32, 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)."

14.5.3.6.6, 14-52 In Table 14-37, 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:".

14.5.3.6.12, 14-55 Delete the first sentence and replace Figure 14-45 with the following figure (exposing the Link Fail status field):



14.5.3.6.12, 14-56 In Table 14-43, 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

14.5.3.6.12, 14-55 In Figure 14-45, correct the reset value for the IFSTAT register to be 0x0000_0000.

14.5.3.7.44, 14-79 Correct the first sentence to read:
"Carry register bits are cleared when written with a one."

14.5.3.7.45, 14-80 Correct the first sentence to read:
"Carry register bits are cleared when written with a one."

14.5.3.7.47, 14-83 In Figure 14-94, change bit 30 to be “Reserved”.

14.5.3.8.1, 14-84 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.”

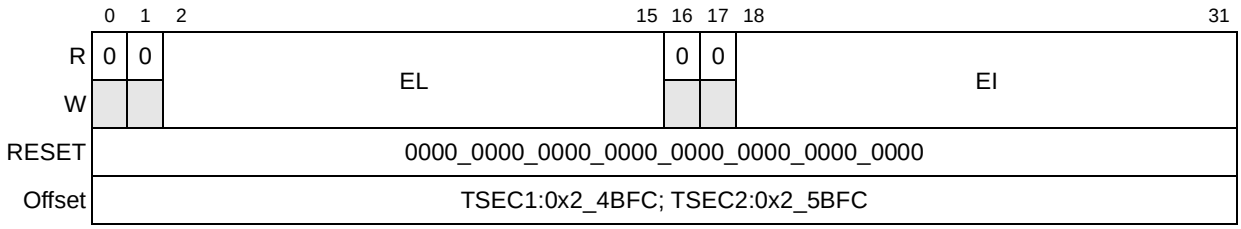
14.5.3.8.1, 14-85 In Table 14-93, add the following sentence to the IADDR*n* field description:

“For instance, the MSB of IADDR0 correlates to entry 0 and the LSB of IADDR7 correlates to entry 255.”

14.5.3.8.2, 14-85 In Table 14-94, 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.”

14.5.3.9.2, 14-87 Replace Figure 14-98 with the following figure to correspond to Table 14-96:



14.6.2.2, 14-108	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”
14.6.2.2, 14-108	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)."
14.6.2.2, 14-109	Correct step 11 of the graceful reset procedure to read: "11. Set WWR and WOP bits in DMACTRL register"
14.6.2.3, 14-109	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."
14.6.2.3, 14-110	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."

14.6.2.6.2, 14-114 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."

14.6.2.6.3, 14-114 Add the following section after Section 14.6.2.6.2:

14.6.2.6.3 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

14.6.2.8, 14-116 In Table 14-115, 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."

14.6.2.11, 14-118 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."

14.6.3, 14-120 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;
};"
```

14.6.3.2, 14-124 In Table 14-120, 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."

14.6.4, 14-125 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."

14.7.1.1, 14-129 In Table 14-123, 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]

14.7.1.2, 14-129 Change the title of Table 14-124 to read "GMII Interface Mode Signal Configuration."

14.7.1.2, 14-129 In Table 14-124, 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.

14.7.1.2, 14-132 In Table 14-126, 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]

14.7.1.3, 14-136 In Table 14-129, 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]

14.7.1.4, 14-137 Change the title of Table 14–130 to read "RGMII Interface Mode Signal Configuration."

14.7.1.4, 14-140 — In Table 14-132, 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]

14.7.1.5, 14-144 — In Table 14-135, 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 15 — Correct the DMA signal enumeration for consistency as in the following example:

"DMA0_DREQ" to "DMA_DREQ0"

15.2.2, 15-5 and

15.4.1.3, 15-32 — 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."

15.3.1, 15-7 — In Table 15-4, 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

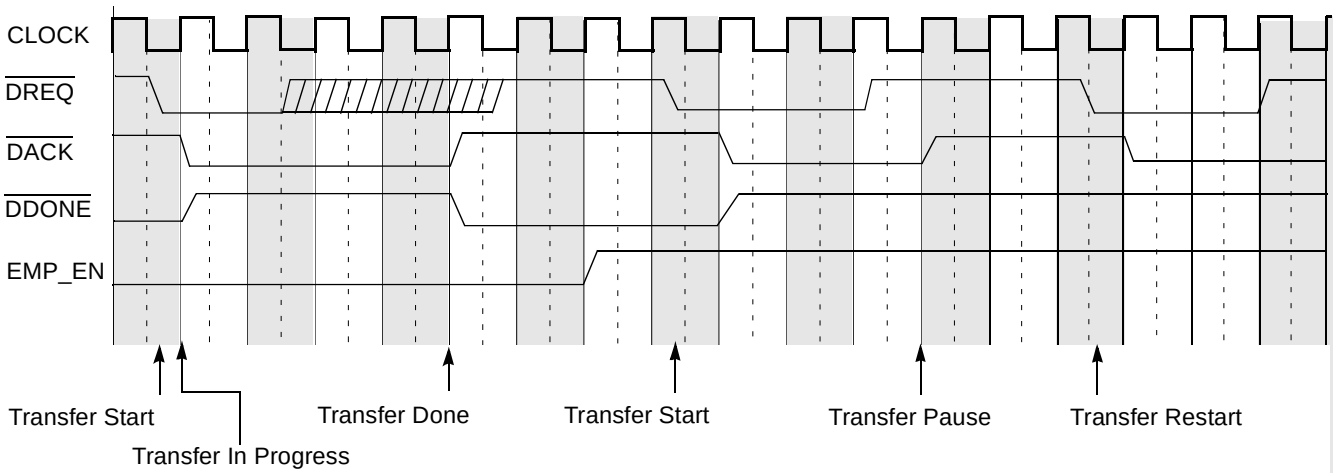
0x2_12AC

0x2_12C8–0x2_12FC

15.3.2.5.1, 15-17 — 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.

15.3.2.7.1, 15-20 — 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.
15.3.2.10, 15-22	In Figure15-17, correct the offset value for CLSDAR2 to 0x2_1234.
15.3.2.11, 15-23	In Figure15-18, correct the offset value for the NLSDAR*n*–DMA 2 next list descriptor address register to 0x2_123C.
15.5.1.1.1, 15-27	Change step 1 in the sequence to read, "Poll the channel state (see Table 15-21 "Channel State Table") to confirm that the specific DMA channel is idle."
15.5.1.1.2, 15-28	Change step 1 in the sequence to read, "Poll the channel state (see Table 15-21 "Channel State Table") to confirm that the specific DMA channel is idle."
15.5.1.1.3, 15-29	Change step 2 in the sequence to read, "Poll the channel state (see Table 15-21 "Channel State Table") to confirm that the specific DMA channel is idle."
15.5.1.1.4, 15-29	Change step 3 in the sequence to read, "Poll the channel state (see Table 15-21 "Channel State Table") to confirm that the specific DMA channel is idle."
15.5.1.2.3, 15-30	Change step 2 in the sequence to read, "Poll the channel state (see Table 15-21 "Channel State Table") to confirm that the specific DMA channel is idle."
15.5.1.2.4, 15-31	Change step 3 in the sequence to read, "Poll the channel state (see Table 15-21 "Channel State Table") to confirm that the specific DMA channel is idle."
15.4.1.3, 15-32	Replace Figure 15-22 with the following, showing more flexible negation of $\overline{\text{DREQ}}$:



15.5.1.2, 15-42	Remove the last sentence of the second paragraph.
16.2, 16-12	In Table 16-2, 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 MPC8540 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)."
16.3.1.2, 16-20	Add the following sentence to the end of the third paragraph: "Note that outbound translation windows must not overlap the configuration access registers."
16.3.1.2.4, 16-22	In Figure 16–9, 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."
16.3.1.2.4, 16-22	In Table 16–10, 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."
16.3.1.2.4, 16–23	In Table 16–10, 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."
16.3.1.4, 16–27	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 16-57 for PCI mode error actions and Table 16-64 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.”
16.3.1.4, 16–28	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.”
16.3.1.4.7, 16–32	Add the following sentence before Figure 16-21: “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.”
16.3.1.4.7, 16–32	Add the following immediately before Figure 16-21: “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).”
16.3.1.4.8, 16-33	Add the following sentence before Figure 16-22: “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.”
16.3.2, 16-35	Insert the following after Figure 16-26: “Note that software must be restricted from accessing any reserved or undefined register space on the MPC8540; doing so may hang the device.”
16.3.2.3, 16-37	In Table 16-27, add the following sentence to the description of the Bus master bit field: “Note that the Bus master bit in the MPC8540’s PCI bus command register must be set before any outbound configuration access is attempted.”
16.3.2.4, 16–37	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 16-57 for PCI mode error actions and Table 16-64 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).”

16.3.2.9, 16-40 In Figure 16-35, change the reset value from “0000_0000” to “0000_0000 (PCI); 0000_1000 (PCI-X)”.

16.3.2.19, 16-47 In Table 16-46, 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”

16.4.1, 16-51 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.”

16.4.1.3, 16-53 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.”

16.4.2.11.1, 16-69 Insert the following between the first and second sentences in the first paragraph:

“Note that the Bus Master bit in the MPC8540’s PCI bus command register must be set before an outbound configuration access is attempted.”

16.4.2.11.2–16.4.2.11.3, 16-69–16-71

Replace the existing sections with the following:

16.4.2.11.2 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 MPC8540’s PCI bus command register must be set before an outbound configuration access is attempted. Device 0 on PCI/X bus 0 is the MPC8540 itself; thus, device 0, bus 0 is used to access the internal PCI/X configuration header.

When the MPC8540 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 MPC8540 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 MPC8540 performs a type 0 configuration cycle translation. If the bus number indicates a remote PCI bus (that is, nonlocal), the MPC8540 performs a type 1 configuration cycle translation. 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 MPC8540 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)
```

16.4.2.11.3 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 MPC8540 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 MPC8540 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.

16.4.2.13, 16-75 Delete the second and third sentences in the first paragraph.

16.4.2.13.1, 16-75 Delete the last paragraph.

16.4.2.13.2, 16-76 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.”

16.4.3.6.4, 16-88 Add the following new section that describes the PCI-X split completion message:

16.4.3.6.4 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 16-78.

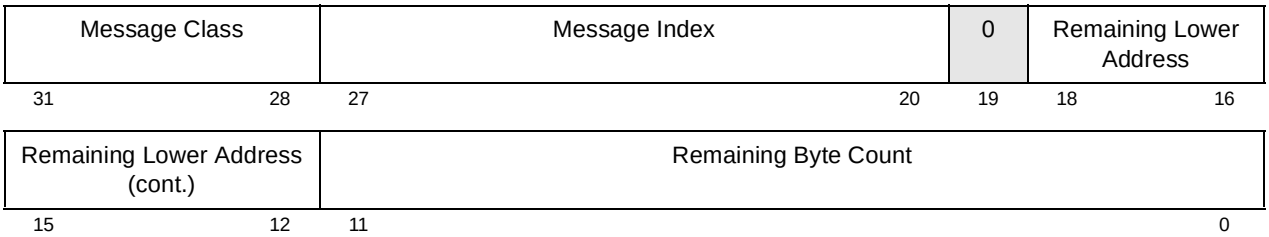


Figure 16-78. PCI-X Split Completion Message Format

Table 16-62 describes the fields of the PCI-X split completion message.

Table 16-62. 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 MPC8540 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.

16.4.3.7, 16-88 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.”
16.4.3.7, 16-88	Add the following before the first sentence: “As is the case for PCI configuration accesses, the Bus master bit in the MPC8540’s PCI bus command register must be set before an outbound configuration access is attempted.”
16.4.3.8.1, 16-91	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.”
16.4.3.8.1, 16–93	Add the following after Table 16-64: “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.”

16.5.4, 16–95 Add the following new section:

16.5.4 PCI-X Inbound Reads that Cross Window Boundaries

For Rev. 1 of the MPC8540, 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 MPC8540 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.

16.5.5, 16–95 Add the following new section:

16.5.5 Bridging Between RapidIO and PCI/X

The MPC8540 does not support bridging from RapidIO according to the RapidIO Interoperability Specification. If an external RapidIO device tries to target the MPC8540 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 MPC8540 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 MPC8540s communicating with each other via PCI or PCI-X may deadlock when two RapidIO devices (one on each MPC8540) are simultaneously performing priority-1 writes to targets on the remote MPC8540 through the PCI/X link between the devices. To support such a topology, RapidIO write transactions that bridge to the other MPC8540 must be priority 0.

17.3, 17-9–2-13 In Table 17-6, correct the access designation for all RapidIO Registers listed as “Special” to “R/W”.

17.3.1.5, 17-16	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."
17.3.1.20, 17-30	In Table 17-26, add the following cross-reference to the field description for PLMREQCSR[C]: “See Table 17-85 for command symbol formats.”
17.4.1, 17-83	In Table 17-83, 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."
17.4.1, 17-86	In Table 17-85, 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."
18.4, 18-4	In Table 18-3, change the reset value of the GPINDR register to be 0x*nnnn*_0000.
18.4.1.9, 18-11	In Figure 18-9, change the reset value of the GPINDR register to be *nnnn_nnnn_nnnn_nnnn*_0000_0000_0000_0000.
18.4.1.14, 18-17	In Table 18-17, change the bit ranges “32–47” to “0–15” and “48–63” to “16–31”.
18.5.1.5.3, 18-23	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.”
19.3.1, 19-4	In Table 19-1, correct the name of the register with offset 0xE_1094 from PMLCB5 to PMLCB8.

20.2.1, 20-7 In Table 20-2, add the following signals:

$\overline{\text{LSSD\_MODE}}$	Test	Test	Factory test. Refer to the *MPC8540 Integrated Processor Hardware Specifications* for proper treatment.		I	20-9
L1_TSTCLK	Test	Test	Factory test. Refer to the *MPC8540 Integrated Processor Hardware Specifications* for proper treatment.		I	20-9
L2_TSTCLK	Test	Test	Factory test. Refer to the *MPC8540 Integrated Processor Hardware Specifications* for proper treatment.		I	20-9

20.2.2.3, 20-9 In Table 20-5, add the following signal descriptions:

$\overline{\text{LSSD\_MODE}}$	I	Used for factory test. Refer to the *MPC8540 Integrated Processor Hardware Specifications* for proper treatment.
$\overline{\text{L1\_TSTCLK}}$	I	Used for factory test. Refer to the *MPC8540 Integrated Processor Hardware Specifications* for proper treatment.
$\overline{\text{L2\_TSTCLK}}$	I	Used for factory test. Refer to the *MPC8540 Integrated Processor Hardware Specifications* for proper treatment.

20.3.1.2, 20-13 Change the last sentence in the first paragraph to, “Note that the transaction address is qualified with the bits described in Section 20.3.1.3, “Watchpoint Monitor Address Mask Register (WMAMR),” before being compared with WMAR. Note also that the contents of WMAR are not qualified with WMAMR.”

20.3.1.3, 20-13 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.”

20.3.2.2, 20-18 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 20.3.2.3, “Trace Buffer Address Mask Register (TBAMR), ” before being compared with TBAR. Note that the contents of TBAR are not qualified with TBAMR.”

20.3.2.3, 20-18 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.”

20.3.2.4, 20-19 In the last sentence of the first paragraph, replace “TBCR[TMD]” with “TBCR0[TMD]”.

20.4.6.1, 20-29 In Table 20-27, 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”

20.4.6.1, 20-30 and 20-31

Change all instances of “TRSEL” and “TRSEL1” to “IFSEL” and all instances of “TBCR” to “TBCR1”.

21.2, 21-5 In the first bullet item of the FEC features list, change “820.3x” to “802.3x.”

21.5.2, 21-10 In Table 21-3, add the following row:

0x2_604C	FIFO_PAUSE_CTRL—FIFO pause control register	R/W	0x0000_0000	14.5.3.2.1/A-19

21.5.2, 21-11 In Table 21-3, change the reset value for the FEC IFSTAT register to be 0x0000_0000.

21.5.3.1.1, 21-14 In Table 21-4, 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.

0 Control frame not received

1 Control frame received”

21.5.3.1.1, 21-14 In Table 21-4, 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.”

21.5.3.1.2, 21-16 Replace the last sentence in the first paragraph with:

"The interrupt signal can be cleared by clearing the corresponding IEVENT bit."

21.5.3.1.3, 21-17 Undocument bitfields TXEDIS, CRL/XDADIS, and XFUNDIS.

21.5.3.1.6, 21-20 Remove all references to bitfield DMACTRL[TOD]. This bit position (29) is now reserved.

21.5.3.2.1, 21-21 Insert the following section as the first subsection of Section 21.5.3.2:

21.5.3.2.1 FIFO Pause Control Register (FIFO_PAUSE_CTRL)

FIFO_PAUSE_CTRL, shown in Figure 21-11, is writable by the user to configure the properties of the FEC FIFO.

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

	16													29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TFC_PAUSE_EN	0
W																
Reset	0000_0000_0000_0000															
Offset	0x2_604C															

Figure 21-11FIFO_PAUSE_CTRL Register Definition

Table 21-10 describes the fields of the FIFO_PAUSE_CTRL register.

Table21-10FIFO_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.

21.5.3.3.1, 21-24 In Table 21-13, 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.”

21.5.3.4.2, 21-31 In Table 21-22, replace the description for the RSTAT[QHLT] field with the following:

“RxBD queue is halted. When IEVENT[BSY] or IEVENT[EBERR] is set during reception of a packet, RSTAT[QHLT] is also set. In order to begin receiving packets again, the user must clear RSTAT[QHLT]. This bit is set whenever the FEC reads an RxBD with its Empty field cleared or encounters a system bus error while processing an RX packet. It is a hardware-initiated stop indication (DMA_CTRL[GRS] being set by the user does not cause this bit to be set.). The current frame and all other frames directed to the halted queue are discarded. A write with a value of 1 re-enables the queue for receiving.

0 RxBD queue is enabled for Ethernet reception. (That is, it is not halted.)

1 All Ethernet controller receive activity to RxBD queue is halted.”

21.5.3.4.2, 21-30 Replace Figure 21-23 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	0
W																																
Reset	0000_0000_0000_0000_0000_0000_0000_0000																															
Offset	0x2_6304																															

21.5.3.6.1, 21-37 In Table 21-28, 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).”

21.5.3.6.1, 21-37 In Table 21-28, 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)."

21.5.3.6.6, 21-41 In Figure 21-34, change the reset value for the FEC IFSTAT register to be 0x0000_0000.

21.5.3.6.6, 21-40 and 21-41

Delete the first sentence and replace Figure 21-35 with the following figure (exposing the Link Fail status field):

	0																				21	22	23				28		31
R	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	Link Fail	0	0	0
W																															
Reset	0000_0000_0000_0000_0000_0000_0000_0000																														
Offset	TSEC1:0x2_453C; TSEC2:0x2_553C																														

21.5.3.6.6, 21-41 In Table 21-34, 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

21.5.3.7.1, 21-43 Change the second 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."

21.5.3.7.1, 21-43 In Table 21-36, 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."

21.5.3.7.2, 21-44 In Table 21-37, 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."

21.6.2.2, 21-48	Replace the soft reset steps with the following: 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.
21.6.2.3, 21-49	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."
21.6.2.3, 21-50	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."
21.6.2.5.2, 21-54	Replace the third and fourth sentences of the first paragraph with the following: “The eight high-order bits of a cyclic redundancy check (CRC) checksum are used to index into the hash table.”

21.6.2.5.3, 21-54 Add the following section after Section 21.6.2.5.2:

21.6.2.5.3 CRC Computation Examples

There are many algorithms for calculating the CRC value of a number. Refer to the RFC 3309 standard, which can be found at http://www.faqs.org/rfcs/rfc3309.html, to compute the CRC value for the purposes of 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

21.6.2.7, 21-56 — In Table 21-44, 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."

21.6.3.2, 21-63 — In Table 21-48, change the LG bitfield description to read:

"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."

21.6.4, 21-64 — Replace the first paragraph with the following:

"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."

21.7.1.1, 21-66 — In Table 21-50, remove the sixth initialization step.

21.7.1.1, 21-66 — In Table 21-50, 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]

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

U

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